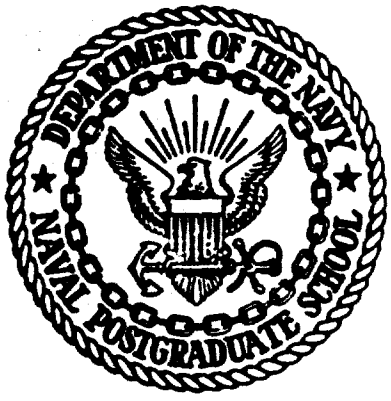
NAVAL POSTGRADUATE SCHOOL
Monterey, California



THESIS

A Look at Some Methods of Solving
Partial Differential Equations
and Eigenvalue Problems

by

Edward Leon Bloxom

Thesis Advisor: Frank D. Faulkner

March 1972

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)	2a. REPORT SECURITY CLASSIFICATION
Naval Postgraduate School Monterey, California 93940	Unclassified
	2b. GROUP

3. REPORT TITLE

A Look at Some Methods of Solving Partial Differential Equations and Eigenvalue Problems

4. DESCRIPTIVE NOTES (Type of report and, inclusive dates)

Master's Thesis; march 1972

5. AUTHOR(S) (First name, middle initial, last name)

Edward Leon Bloxom

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
March 1972	104	10

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
b. PROJECT NO.	
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d.	

10. DISTRIBUTION STATEMENT

This document has been approved for public release and sale; its distribution is unlimited

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Naval Postgraduate School Monterey, California 93940

13. ABSTRACT

Four techniques for the numerical solution of partial differential equations and eigenvalue problems were investigated. Typical problems considered were elliptic partial differential equations of the form

$$U_{xx} + U_{yy} = f(x,y), \qquad (1)$$

or

$$U_{xx} + U_{yy} + \lambda^2 U = 0, \qquad (2)$$

where appropriate boundary conditions are specified so that the problem is self-adjoint.

The four methods are relaxation, Galerkin, Rayleigh-Ritz, and dynamic programming combined with Stodola's method, for eigenvalue problems.

The results indicated that for eigenvalue problems relaxation or dynamic programming modified is to be preferred usually and for partial differential equations Galerkin or dynamic programming is preferred.

DD FORM 1 NOV 1473 (PAGE 1)

S/N 0101-807-6811

Security Classification

A-31408

KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Eigenvalues						
Eigenfunctions						
Dynamic programming						
Galerkin						
Rayleigh-Ritz						
Relaxation						
Partial Differential Equation						

A Look at Some Methods of Solving Partial Differential Equations and Eigenvalue Problems

by

Edward Leon Bloxom
Captain, United States Marine Corps
B.S., Virginia Military Institute, 1965

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE
(with major in mathematics)

from the
NAVAL POSTGRADUATE SCHOOL
March 1972

Author Edward L. Bloxom

Approved by: Frank D. Faulkner
Thesis Advisor

R. E. Gaskell
Chairman, Department of Mathematics

William M. Clarkin
Academic Dean

ABSTRACT

Four techniques for the numerical solution of partial differential equations and eigenvalue problems were investigated. Typical problems considered were elliptic partial differential equations of the form

$$U_{xx} + U_{yy} = f(x,y), \tag{1}$$

or

$$U_{xx} + U_{yy} + \lambda^2 U = 0, \tag{2}$$

where appropriate boundary conditions are specified so that the problem is self-adjoint.

The four methods are relaxation, Galerkin, Rayleigh-Ritz, and dynamic programming combined with Stodola's method, for eigenvalue problems.

The results indicated that for eigenvalue problems relaxation or dynamic programming modified is to be preferred usually and for partial differential equations Galerkin or dynamic programming is preferred.

TABLE OF CONTENTS

LIST OF TABLES

I. INTRODUCTION

Initially for this thesis it was planned to investigate methods for finding eigenfunctions and eigenvalues, with a particular interest in the oscillation in basins such as harbors and bays. The report by Angel [10] introduced a new mothod, that of dynamic programming, for the solution of some partial differential equations. This method seemed promising and it was then extended here to the solution of other partial differential equations. It also made it possible to invert a linear differential operator and hence apply Stodola's method in finding eigenvalues and eigenfunctions. This led naturally to a comparison with several procedures already known to try to compare convergence, speed, and accuracy, by applying them to the solution of several rather simple problems.

It is assumed that the reader has some familiarity with the techniques of replacing differential equations by difference equations and the standard technique of separating variables in linear partial differential equations. The regions and their boundaries were assumed to be "nice," not excessively irregular. Some knowledge of calculus of variations also is desirable but not necessary; reference [5] provides more than adequate background.

Four problems were divided into two categories. In the first category were eigenvalue problems. Two typical ones were

$$U_{xx} + U_{yy} = -\lambda^2 U \qquad \text{in } A, \qquad (1.1)$$

subject to the constraint

$$U(x,y) = 0 \qquad \text{on } \delta A, \qquad (1.2)$$

where δA is the boundary of the domain A; and second

$$\nabla^4 U = \lambda^2 U \qquad \text{in } A, \qquad (1.3)$$

subject to the constraint

$$U(x,y) = \frac{\partial^2 U}{\partial^2 n^2} = 0 \qquad \text{on } \delta A. \qquad (1.4)$$

In the second category were equations such as Poisson's equation

$$U_{xx} + U_{yy} = f(x,y) \qquad \text{in } A, \qquad (1.5)$$

subject to the constraint

$$U(x,y) = g(x,y) \qquad \text{on } \delta A, \qquad (1.6)$$

and the biharmonic equation

$$\nabla^4 U = f(x,y) \qquad \text{in } A, \qquad (1.7)$$

subject to the constraints

$$U(x,y) = h(x,y) \qquad \text{on } \delta A \qquad (1.8)$$

and

$$\frac{\partial^2 U}{\partial n^2} = q(x,y) \qquad \text{on } \delta A. \qquad (1.9)$$

Problems in the first category were numerically solved by a relaxation method, the Rayleigh-Ritz method, and dynamic programming combined with Stodola's method.

Problems in the second category were solved by Galerkin's method, the Rayleigh-Ritz method and dynamic programming.

The domain used throughout this paper for comparisons was the unit square, with the constraint

$$U(x,y) = 0 \qquad \text{on } \delta A. \qquad (1.10)$$

Computations were also carried out for L-shaped and triangular regions and for other boundary conditions.

In the relaxation method and dynamic programming method in solving eigenvalue problems the initial estimates to the eigenfunctions had a special property. The first approximation to U_1 was picked so that it had no negative value in the domain. The approximation to U_2 was picked so that it had negative and positive values in the domain and in addition it was orthogonal to U_1. The initial approximations to U_3 was picked in the same way as the one for U_2 except that it had to be orthogonal to both U_1 and U_2. It may be difficult to make a suitable choice for some of the higher modes.

II. RELAXATION METHOD

For many years relaxation techniques have been used to solve differential equations with and without the aid of computers. They are basically iterative procedures in which a new approximation is obtained from a previous approximation and its residuals.

A. DERIVATION OF EQUATIONS

In this section a typical problem is posed and solved by a relaxation method.

Suppose the problem to be solved has the following form

$$Z_{tt} = Z_{xx} + Z_{yy} \qquad \text{in } A, \qquad (2.1)$$

where the unknown function Z must satisfy the differential equation in a simply connected region A in the xy plane, and for $t>0$. In addition the function Z is required to vanish at points on the boundary δA of the region A,

$$Z(x,y,t) = 0 \qquad \text{on } \delta A. \qquad (2.2)$$

A typical problem is that of a vibrating membrane. The function $Z(x,y,t)$ denotes the vertical displacement of the membrane.

Now assume that the displacement has a representation of the form

$$Z(x,y,t) = T(t)\, U(x,y). \qquad (2.3)$$

When Eq. (2.3) is combined with Eq. (2.1) the variables may be separated and a new equation is obtained of the form

$$\frac{T''}{T} = \frac{(U_{xx} + U_{yy})}{U} = -\lambda^2 . \tag{2.4}$$

This is equivalent to two equations

$$T'' + \lambda^2 T = 0 \tag{2.5}$$

and

$$U_{xx} + U_{yy} = -\lambda^2 U, \tag{2.6}$$

each with a parameter λ. Further U must satisfy the boundary condition

$$U(x,y) = 0 \qquad \text{on } \delta A. \tag{2.7}$$

This is a typical eigenvalue and eigenfunction problem: to find the values λ, or λ_n, and the associated functions U, or U_n, satisfying Eq. (2.6) and the constraint (2.7).

The values λ_n, $\lambda_1 \leq \lambda_2 \leq \lambda_3 \leq \cdots$, are called the eigenvalues. With each eigenvalue is an eigenfunction U_n. In this problem the eigenfunctions associated with different eigenvalues are orthogonal on the region A. Each eigenfunction U_n is also called a mode.

Consider a thin elastic membrane of a particular form, such as a very thin uniform sheet of rubber. Assume that the membrane is made fast at the boundary, while it is tightly stretched over the region with uniform tension. Also assume that damping is negligible. Then if an interior region of the membrane is pushed in a direction perpendicular

to the plane of equilibrium, it becomes distorted into a curved surface. The resulting area can be computed as

$$S = \iint_A \sqrt{1 + U_x^2 + U_y^2}\, dxdy, \tag{2.8}$$

Assume that U_x and U_y are very small; Eq. (2.8) becomes approximately

$$S = \iint_A (1 + \tfrac{1}{2}U_x^2 + \tfrac{1}{2}U_y^2)\, dxdy. \tag{2.9}$$

The increase in the area of the membrane due to the distortion is therefore approximated by

$$\begin{aligned} \delta S &= \iint_A (1 + \tfrac{1}{2}U_x^2 + \tfrac{1}{2}U_y^2)\, dxdy - \iint_A dxdy \\ &= \tfrac{1}{2} \iint_A (U_x^2 + U_y^2)\, dxdy. \end{aligned} \tag{2.10}$$

Hence the potential energy of the membrane in the deflected position is

$$PE = v/2 \iint_A (U_x^2 + U_y^2)\, dxdy, \tag{2.11}$$

where v is the tension, assumed to be constant over the region.

Now consider any particular eigenfunction or mode. It follows from the solution of Eq. (2.4) that the deflection is a periodic function of time and may be expressed in the form

$$Z(x,y,t) = U(x,y) \sin \lambda t, \tag{2.12}$$

except for a phase shift. Thus Eq. (2.11) can be rewritten

as

$$PE = \tfrac{1}{2} \iint_A (U_x^2 + U_y^2)\, dxdy \quad \sin^2 \lambda t. \tag{2.13}$$

The maximum value of the potential energy is

$$PE_{max} = v/2 \iint_A (U_x^2 + U_y^2)\, dxdy. \tag{2.14}$$

The kinetic energy of an element $dm = \rho dxdy$ of the membrane is

$$\tfrac{1}{2}\rho dxdy(U_t)^2 = \tfrac{1}{2}\rho dxdy(U^2\lambda^2 \cos^2\lambda t), \tag{2.15}$$

where ρ denotes the mass per unit area of the membrane. Therefore, the kinetic energy of the vibrating system is

$$KE = \tfrac{1}{2}\lambda^2\rho \iint_A U^2 dxdy \cos^2\lambda t, \tag{2.16}$$

and the maximum value of the kinetic energy is

$$KE_{max} = \tfrac{1}{2}\lambda^2\rho \iint_A U^2 dxdy. \tag{2.17}$$

If it is assumed that the energy is constant then, the maximum values of the potential and kinetic energy are equal for individual modes, and therefore

$$\tfrac{1}{2}\lambda^2\rho \iint_A U^2 dxdy = v/2 \iint_A (U_x^2 + U_y^2)\, dxdy \tag{2.18}$$

or,

$$\lambda^2 = \frac{v \iint_A (U_x^2 + U_y^2)\, dxdy}{\rho \iint_A U^2 dxdy} \tag{2.19}$$

The first eigenfunction U_1 is the function which minimizes this quotient and the first eigenvalue λ_1^2 is the corresponding value for the quotient. The next eigenfunction is the function U_2 which minimizes the quotient in the space of functions orthogonal to U_1, and λ_2^2 is the corresponding value of the quotient. The third eigenfunction U_3 minimizes the quotient in the space of functions orthogonal to U_1 and U_2, etc. The set U_1, U_2, ... is unique except that if two or more eigenfunctions have the same eigenvalue, they may be replaced by linear combinations of themselves. It is seen that $\lambda_1 \leq \lambda_2 \leq \lambda_3 \ldots$.

If the mode U is known, Eq. (2.19) can be used to obtain λ^2. An interesting fact is that a rather poor approximation to the first mode U_1, chosen to satisfy the appropriate boundary condition, will yield a surprisingly good estimate for λ_1^2. This result is apparently due to Lord Rayleigh, and the quotient is sometimes called Rayleigh's quotient.

B. COMPUTATIONAL ROUTINE

In this section the results of section A will be used to develope a method to solve Eq. (2.6).

If Eq. (2.6) is expressed as a difference equation then it may be written as

$$\frac{(U_{i+1,j} + U_{i-1,j} - 2U_{i,j})}{h^2} + \frac{(U_{i,j+1} + U_{i,j-1} - 2U_{i,j})}{k^2}$$

$$= -\lambda^2 U_{i,j}, \qquad (2.20)$$

where h and k denote the x and y mesh size respectively. If the mesh sizes are equal then Eq. (2.20) can be rewritten as

$$U_{i,j} = \frac{(U_{i+1,j} + U_{i,j+1} + U_{i-1,j} + U_{i,j-1})}{4 - \lambda^2 h^2} \qquad (2.21)$$

The procedure used to solve the partial differential equation was to pick a function U^o which satisfied the given boundary conditions as a first approximation to U(x,y). This function need not be a very good approximation to U, and in fact step functions were sometimes used.

From Eq. (2.19) an approximation to λ^2 was obtained by assuming that the approximation U^o was the desired function U. With this first approximation to λ^2 Eq.(2.21) was used to obtain an improved estimate of U. The form of Eq. (2.21) used to obtain the νth approximation was

$$U^{\nu}_{i,j} = \frac{(U^{\nu-1}_{i+1,j} + U^{\nu-1}_{i,j+1} + U^{\nu}_{i-1,j} + U^{\nu}_{i,j-1})}{(4 - \lambda^2 h^2)}, \qquad (2.22)$$

in which the ν-1st estimate of λ was used. By alternating Eq. (2.19) and Eq. (2.22) a close approximation to λ^2 and U were obtained. The solution converged to the smallest eigenvalue λ_1 and the associated eigenvector U_1.

C. HIGHER ORDER EIGENVALUES

In this section the method is extended to find the larger eigenvalues and eigenfunctions.

To obtain the larger eigenvalues and eigenvectors the same procedure as that in section B was used except that

additional equations were added to force the eigenfunctions to be orthogonal to those already found. Define U_i for $i = 1, 2, \ldots, n$ to be the ith eigenfunction, associated with λ_i. All higher eigenfunctions must be orthogonal to the lower ones obtained. Orthogonalization was accomplished by subtracting out multiples of the lower eigenfunctions already obtained. This was effected by expressing U as

$$U_{i+1_{new}} = U_{i+1_{old}} - \sum_{j=1}^{n} C_j U_j, \tag{2.23}$$

where

$$C_j = \frac{\iint_A U_{i+1_{old}} U_j \, dxdy}{\iint_A U_j^2 \, dxdy}, \quad j=1,2,\ldots,i. \tag{2.24}$$

For each eigenfunction desired a different initial approximating function was used. The method gave the eigenvalues and associated eigenfunctions in numerically increasing order. The method was very simple and effective for lower eigenvalues.

The fault of the method is that convergence was poor and computation times were large if the number of intervals in both directions was large.

III. DYNAMIC PROGRAMMING [8,10]

The method of dynamic programming has recently been applied to the solution of partial differential equations, [10]. The difference equation may be regarded as leading to a system of linear equations with a large number of unknowns. The method effectively reduces the number of unknowns involved so that a number of systems are to be solved, each one of much lower dimension. In one case, for example, using a grid with N+1 intervals in the region in each direction, N systems each with N unknowns are solved, rather than one system with N - squared unknowns.

In section A, the method is applied to the solution of Poisson's equation over a rectangle, following the paper of Angel [10]. In section B, the method is extended to a related eigenvalue problem by combining it with Stodola's method. In section C, the method is extended to the solution of the biharmonic equation, by applying dynamic programming twice. Finally, in section D, the method is applied to the eigenvalue program involving $\nabla^4 U = \lambda^2 U$, again by combining the method with Stodola's method.

A. DYNAMIC PROGRAMMING FOR PARTIAL DIFFERENTIAL EQUATIONS

Consider the solution of Poisson's equation

$$U_{xx} + U_{yy} = f(x,y) \qquad \text{in A,} \qquad (3.1)$$

where $U = U(x,y)$ is subject to given boundary conditions

such as

$$U(x,y) = g(x,y) \qquad \text{on } \delta A. \tag{3.2}$$

It may be noted that Eq. (3.1) is the Euler equation associated with the variational problem

$$I(U) = \underset{U}{\text{Min}} \iint_A (U_x^2 + U_y^2 + 2fU)\, dxdy \tag{3.3}$$

where the function U is chosen from the class of functions with first partial derivatives belonging to L^2 over A, and satisfying the boundary conditions of (3.2) on δA.

Let the region A be discretized by choosing n+1 and m+1 equally spaced points in the independent variables x and y respectively. Then Eq. (3.3) may be rewritten in a discrete version with equal intervals as

$$I(U) = \underset{U_{i,j}}{\text{Min}} \sum_{i=1}^{n} \sum_{j=1}^{m} \Big[(U_{i,j} - U_{i,j-1})^2 + (U_{i,j} - U_{i-1,j})^2 + 2f_{i,j} U_{i,j} h^2 \Big], \tag{3.4}$$

where $\{U_{o,j}\}$, $\{U_{i,o}\}$, $\{U_{n,j}\}$, and $\{U_{i,m}\}$ are determined from the boundary conditions of Eq. (3.2). Now, if all terms in Eq. (3.4) involving only boundary values were removed, while not affecting the solution, a more convenient form of Eq. (3.4) is obtained

$$I(U) = \underset{U_{i,j}}{\text{Min}} \sum_{i=1}^{n} \Bigg[\sum_{j=1}^{m} (U_{i,j} - U_{i,j-1})^2 + \sum_{j=1}^{m-1} 2h^2 f_{i,j} U_{i,j} + \sum_{j=1}^{m-1} (U_{i,j} - U_{i-1,j})^2 \Bigg] \tag{3.5}$$

In vector notation Eq. (3.5) may be rewritten in the form

$$I(U) = \underset{U_R}{\text{Min}} \sum_{i=1}^{n} (\langle \bar{Q}\bar{U}_R, \bar{U}_R \rangle + \langle \bar{r}_R, \bar{U}_R \rangle + S_R \qquad (3.6)$$

$$+ \langle \bar{U}_R - \bar{U}_{R-1}, \bar{U}_R - \bar{U}_{R-1} \rangle + \langle \bar{f}_R, \bar{U}_R \rangle)$$

In this, $\bar{U}_R = (U_{R_1}, \ldots, U_{R_{m-1}})^T$. This relation now defines a symmetric matrix $\bar{Q}$, vectors $\bar{r}_R$, and $\bar{f}_R$, and a scalar S_R by

$$\bar{Q} = \{q_{i,j}\}, \text{ where } q_{i,j} = \begin{cases} 2 & i = j \\ -1 & |i-j| = 1 \\ 0 & \text{otherwise} \end{cases} \qquad (3.7)$$

$$\bar{r}_R = \{r_{R,j}\}, \text{ where } r_{R,j} = \begin{cases} -2U_{R,o} & j = 1 \\ -2U_{R,m} & j = m-1 \\ 0 & \text{otherwise} \end{cases}$$

$$\bar{f}_R = \{2h^2 f_{R,j}\}$$

$$S_R = U_{R,o}^2 + U_{R,m}^2 .$$

The notation as in $\langle \bar{r}_R, \bar{U}_R \rangle$, denotes the scalar product of the two corresponding vectors. The matrix $\bar{Q}$ remains constant while $\bar{r}_R$ and S_R are functions of the boundary conditions only.

Now in order to solve Eq. (3.6) a sequence of dynamic programming problems was considered. Let

$$F_R(\bar{V}) = \underset{\bar{U}_R \ldots \bar{U}_{n-1}}{\text{Min}} \sum_{i=R}^{n} (\langle \bar{Q}\bar{U}_i, \bar{U}_i \rangle + \langle \bar{r}_i, \bar{U}_i \rangle \qquad (3.8)$$

$$+ S_i + \langle \bar{U}_i - \bar{U}_{i-1}, \bar{U}_i - \bar{U}_{i-1} \rangle + \langle \bar{f}_i, \bar{U}_i \rangle)$$

where $\bar{U}_{R-1} = \bar{V}$ and $\bar{U}_n$ is given by Eq. (3.2). Now Eq. (3.8) can be rewritten as

$$F_R(\bar{V}) = \min_{\bar{U}_R} (\langle \bar{Q}\bar{U}_R, \bar{U}_R \rangle + \langle \bar{r}_R, \bar{U}_R \rangle + S_R \tag{3.9}$$

$$+ \langle \bar{U}_R - \bar{V}, \bar{U}_R - \bar{V} \rangle + \langle \bar{f}_R, \bar{U}_R \rangle + \langle \bar{Q}\bar{U}_{R+1}, \bar{U}_{R+1} \rangle$$

$$+ \ldots + S_n + \langle \bar{V}_n - \bar{U}_{n-1}, \bar{U}_n - \bar{U}_{n-1} \rangle + \langle \bar{f}_n, \bar{U}_n \rangle).$$

This may be done since the minimization over $\bar{U}_{R+1}, \ldots, \bar{U}_{n-1}$ can be commuted with the minimization over $\bar{U}_R$. This is a common technique of dynamic programming based upon the principle of optimality, [8]. This allows Eq. (3.9) to be rewritten as

$$F_R(\bar{V}) = \min_{\bar{U}_R} \Big[\langle \bar{Q}\bar{U}_R, \bar{U}_R \rangle + \langle \bar{r}_R, \bar{U}_R \rangle + S_R \tag{3.10}$$

$$+ \langle \bar{U}_R - \bar{V}, \bar{U}_R - \bar{V} \rangle + \langle \bar{f}_R, \bar{U}_R \rangle + F_{R+1}(\bar{U}_R) \Big].$$

The final equation is

$$F_n(\bar{V}) = \langle \bar{Q}\bar{U}_n, \bar{U}_n \rangle + \langle \bar{r}_n, \bar{U}_n \rangle + S_n \tag{3.11}$$

$$+ \langle \bar{U}_n - \bar{V}, \bar{U}_n - \bar{V} \rangle + \langle \bar{f}_n, \bar{U}_n \rangle,$$

and $\bar{U}_n$ is known from the boundary conditions Eq. (3.2). Since $F_R(\bar{V})$ is quadratic in $\bar{V}$ Eq. (3.11) may be rewritten in the form

$$F_R(\bar{V}) = \langle \bar{A}_R\bar{V}, \bar{V} \rangle + \langle \bar{b}_R\bar{V} \rangle + C_R. \tag{3.12}$$

Now by substituting from Eq. (3.12) into Eq. (3.10) and then differentiating with respect to $\bar{U}_R$ an expression for

$\bar{U}_R$ is obtained of the form

$$\bar{U}_R = (\bar{I} + \bar{Q} + \bar{A}_{R+1})^{-1} \left(\bar{V} - \frac{\bar{b}_{R+1} + \bar{r}_R + \bar{f}_R}{2} \right). \quad (3.13)$$

This is obtained by substituting related Eq. (3.13) into Eq. (3.10) and then combining Eq. (3.10) with Eq. (3.12), the various quantities in Eq. (3.13) are defined by these steps as

$$\bar{A}_R = \bar{I} - (\bar{I} + \bar{Q} + \bar{A}_{R+1})^{-1} \quad (3.14)$$

$$\bar{b}_R = (\bar{I} + \bar{Q} + \bar{A}_{R+1})^{-1} (\bar{b}_{R-1} + \bar{r}_R + \bar{f}_R) \quad (3.15)$$

$$C_R = C_{R+1} + S_R - \left[(I + Q + A_{R+1})^{-1} \right. \quad (3.16)$$
$$\left. < \frac{\bar{b}_{R+1} + \bar{r}_R}{2}, \frac{\bar{b}_{R+1} + \bar{r}_R}{2} > \right]$$

with initial values determined from Eq. (3.11) as

$$\bar{A}_n = \bar{I}, \quad (3.17)$$

$$\bar{b}_N = -2\bar{U}_n, \quad (3.18)$$

$$C_n = <(\bar{I} + \bar{Q})\, \bar{U}_n, \bar{U}_n> + <\bar{r}_n - \bar{U}_n> + S_n. \quad (3.19)$$

The matrix $(\bar{I} + \bar{Q} + \bar{A}_{R+1})$ is nonsingular, 1 . Thus it has inverses which may be computed beforehand, since it is dependent only upon the type of operator.

Due to the fact that only the values of $\bar{U}_R$ are desired the quantities C_R need not be calculated.

The procedure is to calculate the quantities in Eq. (3.14) repeatedly until $\bar{A}_2$ and $\bar{b}_2$ are obtained. Then

$\bar{U}$ is found from Eq. (3.13) with $\bar{V} = \bar{U}_o$, which is known from the boundary conditions. Next, Eq. (3.13) is repeatedly solved using the stored values of $\bar{A}_R$ and $\bar{b}_R$, and the last value of $\bar{U}_R$ as $\bar{V}$.

Thus, the problem was solved by n-1 inversions of symmetric matrices of order m-2. While these matrices may be large there are efficient computer routines available to determine the inverses. Once the inverses are found, they may be stored for future use, since they are based only on the geometry of the region A. Thus for several problems over the same region the inverses may be entered into the program as data. Also they have the property that if less than n+1 grid points are required a reduced number of the matrices may be used.

B. DYNAMIC PROGRAMMING FOR EIGENVALUE PROBLEMS

In the first section of this chapter it was seen that dynamic programming could be used to solve partial differential equations. In this section, the program is modified to solve a related eigenvalue problem by Stodola's method.

Consider the problem of the vibrating membrane considered in Chapter II. The differential equation for the eigenvalues and eigenfunctions is

$$U_{xx} + U_{yy} = -\lambda^2 U \qquad \text{in A,} \qquad (3.20)$$

where $U = U(x,y)$ is subject to the boundary condition

$$U(x,y) = 0 \qquad \text{on } \delta A. \qquad (3.21)$$

This problem was solved by two related methods. In both Eq. (3.20) is regarded as a special case of Eq. (3.1) in which

$$f(x,y) = -\lambda^2 U(x,y) \qquad \text{in } A \qquad (3.22)$$

and

$$f(x,y) = 0 \qquad \text{on } \delta A \qquad (3.23)$$

1. First Routine

The difficulty here was that the function f was known only on the boundary and λ was unknown. The first step, to overcome this obstacle, was to choose a function $U^o(x,y)$ which satisfied the given boundary conditions. Then an estimate of λ^2 was obtained, say $(\lambda^o)^2$, by using Rayleigh's formula. Next a new estimate of U, say $U^1(x,y)$, was obtained using dynamic programming to solve the equation

$$U^1_{xx} + U^1_{yy} = -(\lambda^o)^2 \, U^o = f^o(x,y). \qquad (3.24)$$

By repeating this sequence of Rayleigh's formula and dynamic programming, a good approximation to the minimum eigenvalue and the associated eigenfunction was obtained. The next two eigenvalues and vectors were also obtained by the process of forcing the higher eigenfunctions to be orthogonal to ones already obtained as was done in relaxation. The inverse matrices used in the routine were calculated in determining the first eigenvalue and eigenfunction; they were then stored so that it was not necessary to recalculate them for subsequent eigenvalues.

2. Stodola's Method

The second form is more like the usual form of Stodola's method and hence a brief of Stodola's technique is given first.

An initial function V_0 satisfying the boundary conditions is selected. It may be considered to be of the following form

$$V_0 = a_1U_1 + a_2U_2 +... \tag{3.25}$$

where a_1 is not equal to zero. For convenience V_0 was normalized; the L_∞ norm of V_0, $||V_0||$, is defined as

$$\underset{A}{MAX}\ |V_0(x,y)| = ||V_0|| ; \tag{3.26}$$

and $||V_0||$ is set equal to one. Now consider

$$L^{-1}V_0 = \frac{-a_1}{\lambda_1^2} U_1 - \frac{a_2}{\lambda_2^2}, U_2 -... \tag{3.27}$$

and

$$L^{-m}V_0 = \left(\frac{-1}{\lambda_1^2}\right)^m \left[a_1U_1 + a_2\left[\frac{\lambda_1}{\lambda_2}\right]^{2m} U_2 +...\right] \tag{3.28}$$

In Eq. (3.28) it can be seen that the relative size of the components U_2, U_3,... are decreasing by a factor $(\lambda_1/\lambda_2)^2$, $(\lambda_1/\lambda_3)^2$,... respectively. After a few applications of the operator, L^{-1}, to V_0, the leading term will dominate. Thus the functions obtained will approximate V_1, except for a constant factor, and the ratio of successive iterates will approach a constant, $-1/\lambda^2$.

3. Second Routine

This was applied as follows. Let Z_m be defined by the equation $LZ_m = V_{m-1}$, so that

$$Z_m = L^{-1}V_{m-1}. \tag{3.29}$$

This equation was solved by dynamic programming. It was found convenient to normalize each iteration. Let

$$V_m = \frac{-Z_m}{||Z_m||} . \tag{3.30}$$

Then the approximation can be made

$$\lambda_1^2 = \frac{1}{||Z_m||} . \tag{3.31}$$

The sequence of functions V_0, V_1,... converges to U_1.

The process was terminated when successive approximations were sufficiently close together, say,

$$||V_m(x,y) - V_{m-1}(x,y)|| < \varepsilon, \tag{3.32}$$

where ε is a preassigned small positive number. The resulting function V_m is an approximation to U_1, and λ_1 is obtained from Eq. (3.31).

C. DYNAMIC PROGRAMMING FOR A HIGHER-ORDER OPERATOR

In section A and B of this chapter it was shown how dynamic programming could be used to solve Poisson's equation and the vibrating membrane program, respectively. In this section, by another modification to the routine, the biharmonic equation can be solved.

Assume the problem to be solved is of the form

$$\nabla^4(U) = f(x,y) \qquad \text{in } A, \tag{3.33}$$

where $U = U(x,y)$ is subject to the constraints

$$\begin{cases} U = g(x,y) & \text{on } \delta A \\ \partial^2 U/\partial n^2 = P(x,y) & \text{on } \delta A. \end{cases} \tag{3.34}$$

Clearly Eq. (3.33) may be rewritten in the form

$$\nabla^2\, \Phi(x,y) = f(x,y), \tag{3.35}$$

where

$$\nabla^2\, U(x,y) = \Phi(x,y). \tag{3.36}$$

Now, since U and $\partial^2 U/\partial n^2$ are both known as the boundary, $\Phi(x,y)$ may be approximated on the boundary. On a rectangle, for example, the following relations determine Φ on the boundary

$$\begin{cases} \Phi(0,y) = -P(0,y) + U_{yy}\ (0,y) \\ \Phi(n,y) = P(n,y) + n_{yy}\ (n,y) \\ \Phi(x,0) = -P(x,0) + U_{xx}\ (x,0) \\ \Phi(x,m) = P(x,m) + U_{xx}\ (x,m). \end{cases} \tag{3.37}$$

The usual finite difference scheme may be used to approximate U_{xx} and U_{yy} and thus the relations of Eq. (3.37) may be approximated by

$$\Phi_{0,j} = -P_{0,j} + \frac{(U_{0,j+1} - 2U_{0,j} + U_{0,j-1})}{h^2} \tag{3.38}$$

$$\begin{cases} \Phi_{n,j} = P_{n,j} + \dfrac{(U_{n,j+1} - 2U_{n,j} + U_{n,j-1})}{h^2} \\ \Phi_{i,0} = -P_{i,0} + \dfrac{(U_{i+1,0} - 2U_{i,0} + U_{i-1,0})}{h^2} \\ \Phi_{i,m} = P_{i,m} + \dfrac{(U_{i+1,m} - 2U_{i,m} + U_{i-1,m})}{h^2} \end{cases} \qquad (3.38)$$

$$\text{for } i = 1,\ldots, n-1,\ j = 1,\ldots, m-1,$$

and $\Phi_{0,0}$, $\Phi_{n,0}$, $\Phi_{0,m}$, $\Phi_{n,m}$ are known from the boundary conditions. Thus $\Phi(x,y)$ is now approximated on the boundary. First Eq. (3.35) is solved by dynamic programming to obtain the function Φ in the region. Then Eq. (3.36) is solved by dynamic programming to obtain the desired solution.

D. DYNAMIC PROGRAMMING FOR HIGHER ORDER EIGENVALUE PROBLEMS

This method may be extended to the corresponding eigenvalue and eigenfunction problem much as before. One such physical problem is that of a vibrating uniform plate with hinged edges.

Assume that the differential equation has the form

$$\nabla^4 U = \lambda^2 U, \qquad \text{in } A, \qquad (3.39)$$

and the boundary conditions are

$$U = \frac{\partial^2 U}{\partial n^2} = 0 \qquad \text{on } \delta A \qquad (3.40)$$

The technique of sections B and C may be applied in two steps to the solution of the problem. Let Φ be defined as

$$\Phi = \nabla^2 U. \qquad (3.41)$$

Then the two equations to be solved are

$$\nabla^2\phi^n = \lambda^2 U^n \tag{3.42}$$

and

$$\nabla^2 U^{n+1} = \phi^n, \tag{3.43}$$

subject to the boundary conditions, Eq. (3.40).

In order to start the routine an initial estimate for the function $U(x,y)$, say $\{U_{ij}\}$, is made. The value of λ is estimated as was done in section B. Then Eq. (3.42) and Eq. (3.43) are solved by dynamic programming to get the next approximation $\{U_{i,j}\}$. The same criterion for stopping is used as that in section B.

IV. RAYLEIGH-RITZ METHOD [4,5,6,9]

The Rayleigh-Ritz method has been used for many years to obtain approximations to the solution of partial differential equations and eigenfunction problems. In this method the problem is posed as a minimization problem, say involving an integral. Then some linearly independent functions which satisfy the boundary conditions are chosen. The solution is approximated by a linear combination of these. Finally the coefficients in the approximation are chosen so as to effect minimization. This leads to an eigenvalue problem involving symmetric matrices.

The functions chosen may be, for example, polynomials of low degree, or trigonometric functions. Assume that homogeneous boundary conditions are given. Let $\phi_k = \phi_k(x,y)$ be n functions which satisfy these and approximate the solution U by

$$U(x,y) = \sum_{k=1}^{n} C_k \phi_k. \tag{4.1}$$

A. PARTIAL DIFFERENTIAL EQUATIONS

In this section the solution of

$$U_{xx} + U_{yy} = f(x,y) \qquad \text{in A,} \tag{1.3}$$

is again considered. It is the Euler equation associated with minimizing the integral

$$\iint_A (U_x^2 + U_y^2 + 2fU)\, dxdy. \qquad (4.2)$$

Substituting from Eq. (4.1) into Eq. (4.2) and integration results in a function which may be written

$$I = I(C_1, \ldots, C_n), \qquad (4.3)$$

for functions of the form (4.1). The minimization of this function and an approximate solution to Eq. (1.3) is thus obtained by solving

$$\frac{\partial I}{\partial C_k} = 0, \qquad k = 1, 2, \ldots, n. \qquad (4.4)$$

The effectiveness of the procedure of course depends upon the choice of the approximating functions $\Phi_k(x,y)$.

B. EIGENVALUE PROBLEMS

The method is also applicable to eigenvalue problems. Consider again the problem in Chapter II of finding eigenvalues and eigenfunctions for the equation

$$U_{xx} + U_{yy} = -\lambda^2 U \qquad \text{in } A \qquad (2.6)$$

subject to the boundary condition

$$U(x,y) = 0 \qquad \text{on } \delta A. \qquad (2.7)$$

In many problems λ_1^2, the lowest eigenvalue, is the minimum of the ratio of two integrals. This fact was shown in Chapter II and Eq. (2.19).

If an approximation to U is chosen as in section A, since each function satisfies the linear homogeneous boundary condition, the sum shown in Eq. (4.1) satisfies it. If

this sum is substituted into Eq. (2.19) the resulting values define an upper bound for λ, for all choices of the C's. Further also, the C's may be chosen to give a least upper bound over the subspace spanned by $\Phi_1, \Phi_2, \ldots, \Phi_n$.

If that approximation for U is substituted into Eq.(2.19) the numerator and denominator become quadratic forms in $(C_1, \ldots, C_n) = \overline{C}$. The numerator has the form

$$\sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} C_i C_j = \overline{C}^t \overline{A}\ \overline{C} \tag{4.5}$$

where

$$a_{ij} = \iint_A (\Phi_{i_x} \Phi_{j_x} + \Phi_{i_y} \Phi_{j_y})\, dxdy, \tag{4.6}$$

and the denominator has the form

$$\sum_{i=1}^{n} \sum_{j=1}^{n} b_{ij} C_i C_j = \overline{C}^t \overline{B}\ \overline{C} \tag{4.7}$$

where

$$b_{ij} = \iint_A \Phi_i\ \Phi_j\, dxdy. \tag{4.8}$$

These define the matrices A and B. The first eigenvalue λ_1 minimizes the quotient in Eq. (2.19) and hence the minimum value of

$$\overline{C}^t \overline{A}\ \overline{C} / \overline{C}^t \overline{B}\ \overline{C}$$

defines the minimum value of the quotient in the subspace spanned by $\Phi_1, \ldots, \Phi_n$.

To find this is equivalent to minimizing the quadratic form

$$\lambda_1^2 = \underset{\bar{C}}{\text{Min}}\ \bar{C}^t \bar{A}\ \bar{C} \tag{4.10}$$

subject to the constraint that the second quadratic form assumes the value one

$$\bar{C}^t \bar{B}\ \bar{C} = 1. \tag{4.11}$$

The problem may be solved in two steps. First find the eigenvalues and eigenvectors of B. Let the eigenvalues of $\bar{B}$ be ω_i^2, i=1, 2,..., n, and the associated normalized eigenvectors V_i. Let $\bar{T}$ be the matrix

$$\bar{T} = (V_1, V_2, \ldots, V_n)\ \text{diag}(1/\omega_1, \ldots, 1/\omega_n). \tag{4.12}$$

Then the transformation

$$\bar{C} = \bar{T}\ \bar{D} \tag{4.13}$$

reduces the constraint (4.11) to the form

$$\bar{C}^t \bar{B}\ \bar{C} = \bar{D}^t \bar{D} = 1. \tag{4.14}$$

condition (4.10) becomes

$$\lambda_1^2 = \underset{\bar{D}}{\text{Min}}\ \bar{D}^t \bar{E}\ \bar{D}, \tag{4.15}$$

where

$$\bar{E} = \bar{T}^t \bar{A}\ \bar{T}, \tag{4.16}$$

subject to the constraint (4.14).

Hence the value of λ_1^2 is the smallest eigenvalue of the matrix $\bar{E}$.

Let $\bar{D}_1$ be the associated normalized eigenvector of $\bar{E}$. Then the corresponding minimizing function has coefficients determined from Eq. (4.13)

$$\bar{C}_1 = \bar{T}\,\bar{D}_1. \tag{4.17}$$

This value of λ_1^2 is an upper bound for the first eigenvalue, the least upper bound in the space of functions spanned by $\Phi_1, \ldots, \Phi_n$. In a similar way the second eigenvalue of $\bar{E}$ furnishes an upper bound to the second eigenfunction, etc.

For simple problems, at least, it seems easy to choose the Φ's so that a good bound for the lowest eigenvalue results. It is not clear how to make good choices to get good estimates for the higher eigenvalues.

V. GALERKIN'S METHOD [4,5,6]

Sometimes the solution to boundary-value problems involving partial differential equations can be obtained by forming an associated functional in the form of a definite integral which is to be made stationary. For a problem of this type, the Rayleigh-Ritz method is often an effective procedure for the determination of an approximate solution. However, in many instances it is difficult to find this functional. In such cases, the Galerkin method is often effective.

A. PARTIAL DIFFERENTIAL EQUATIONS

Consider the linear homogeneous boundary value problem of the form

$$LU(x,y) = f(x,y), \tag{5.1}$$

subject to linear homogeneous boundary conditions. The symbol L stands for a linear differential operator, such as ∇^2.

Suppose that an approximate solution is taken in the form

$$U_n(x,y) = \sum_{k=1}^{n} C_k \phi_k(x,y), \tag{5.2}$$

where the coefficients $C_1, \ldots, C_n$ are constants, to be determined, and, as in the previous chapter, the functions $\phi_1, \ldots, \phi_n$ are picked to satisfy the homogeneous boundary

conditions. The coefficients are dependent upon the number of functions picked and therefore, must be recomputed if a larger number of functions are later chosen. While the function U_n satisfies the boundary conditions, it will not, in general, satisfy Eq. (5.1). Thus there is a residual,

$$LU_n(x,y) - f(x,y) = R_n(x,y), \tag{5.3}$$

which can be viewed as an error or penalty function. Assume the solution for $U(x,y)$ can be expressed by an infinite complete series of these linearly independent functions in the form

$$U(x,y) = \sum_{k=1}^{\infty} c_k \Phi_k(x,y). \tag{5.4}$$

Now $U_n(x,y)$ in Eq. (5.2) represents a sequence of partial sums which approximate $U(x,y)$. Now if the condition is imposed that $L(U_n) - f$ be orthogonal to each function $\Phi_i(x,y)$ on the demain A, the following set of equations are obtained

$$\iint_A \left(L(U_n) - f\right) \Phi_k \, dxdy = 0, \text{ for } k = 1, 2, \ldots, n \tag{5.5}$$

or by use of Eq. (5.3)

$$\iint_A R_n(x,y) \, \Phi_k(x,y) \, dxdy = 0, \text{ for } k = 1, 2, \ldots, n. \tag{5.6}$$

If Eq. (5.6) is to hold as $n \to \infty$ it follows that

$$\lim_{n\to\infty} R_n = 0 \tag{5.7}$$

Let $R(x,y)$ be an arbitrary function satisfying the homogeneous boundary conditions. Since $\{\Phi_k\}$ for $k = 1, 2, \ldots$ forms a complete set of functions, constants $a_1, \ldots, a_n, \ldots$ can be found such that

$$R(x,y) = \sum_{k=1}^{\infty} a_k \Phi_k(x,y). \tag{5.8}$$

Thus,

$$\iint_A \lim_{n\to\infty} R_n(x,y)\, \eta(x,y)\, dxdy = 0, \tag{5.9}$$

for any arbitrary $\eta(x,y)$, so that by the fundamental lemma of variational calculus Eq. (5.7) is true "almost everywhere". Suppose further that $L(U_n) \to L(U)$; then Eq. (5.1) holds, by the use of Eq. (5.7).

Galerkin's method requires that the error function $R_n(x,y)$ be orthogonal to each of the functions Φ_k, or that Eq. (5.5) and Eq. (5.6) hold. Now by substituting Eq. (5.2) into Eq. (5.5) an integral is obtained of the form

$$\iint_A \left[L\left(\sum_{k=1}^{n} C_k \Phi_k \right) - f \right] \Phi_i \, dxdy = 0, \tag{5.10}$$

$$i = 1, 2, \ldots, n.$$

This results in a system of n linear algebraic equations in n unknowns $C_1, \ldots, C_n$. Furthermore, the system is inhomogeneous unless the function $f(x,y)$ is orthogonal to each $\Phi_k(x,y)$. Eq. (5.10) may be rewritten

$$\sum_{j=1}^{n} C_j \iint_A L(\Phi_j)\, \Phi_k \, dxdy = \iint_A f\Phi_k \, dxdy, \qquad (5.11)$$

$$k = 1, 2, \ldots, n.$$

In matrix notation this becomes

$$\bar{B}\, \bar{C} = \bar{F}, \qquad (5.12)$$

where

$$\bar{C} = (C_1\ C_2 \ldots C_n)^T \qquad (5.13)$$

$$\bar{B} = \{b_{ij}\} \quad i = 1, 2, \ldots, n, \qquad (5.14)$$

$$j = 1, 2, \ldots, n$$

$$F = (f_1, \ldots, f_n)^T \qquad (5.15)$$

and

$$b_{ij} = \iint_A L(\Phi_i)\, \Phi_j \, dxdy \qquad (5.16)$$

$$f_i = \iint_A f\Phi_i \, dxdy \qquad (5.17)$$

The constants C_i are obtained by solving the system Eq. Eq. (5.12).

Collocation, a convenient variation. One way to get an approximate solution for the C's is the following. Choose n points of the region A. Evaluate the terms in the integral of Eq. (5.10) at these points. This yields n equations for the unknowns C_k. This method, called collocation, is not generally so accurate but it is quicker than carrying out the integrations to define the coefficients Eq. (5.16) and Eq. (5.17).

Galerkins method is also useful to obtain a direct solution of variational problems. Assume it is desired to minimize a functional of the form

$$I(V) = \iint_A F(x,y\ V,V_x,V_y)\ dxdy. \tag{5.18}$$

It is desired to find an extreme value for the functional subject to the condition that $V(x,y)$ is prescribed on the boundary δA of the domain A. It is known that if the existence of an extremizing function $U(x,y)$ is assumed and that the function F possesses continuous derivatives of the second order with respect to its arguements, there results the condition

$$\iint_A \eta(x,y) \left[F_U - \frac{\partial}{\partial_x} F_{U_x} - \frac{\partial}{\partial_y} F_{U_y}\right] dxdy = 0 \tag{5.19}$$

for an arbitrary function $\eta(x,y)$ which has piecewise continuous derivatives in A and which vanishes on δA. This is derived by considering a function of the form

$$V(x,y) = U(x,y) + \varepsilon\eta(x,y), \tag{5.20}$$

and differentiating with respect to ε. If $\eta(x,y)$ is otherwise arbitrary, then one form of the fundamental lemma of variational calculus requires that

$$F_U - \frac{\partial}{\partial_x} F_{U_x} - \frac{\partial}{\partial_y} F_{U_y} = 0. \tag{5.21}$$

Suppose now that $\eta(x,y)$ is the kth function $\Phi_k(x,y)$ and that an orthogonality requirement is imposed as

$$\iint_A \Phi_k \left(Fu - \frac{\partial}{\partial x} Fu_x - \frac{\partial}{\partial y} Fu_y\right) dxdy = 0, \qquad (5.22)$$

for $k = 1, 2, \ldots, n$.

Finally, assume that the function $U(x,y)$ which effects the minimization can be represented satisfactorily by a finite series

$$U_n(x,y) = \sum_{k=1}^{n} c_k \Phi_k(x,y). \qquad (5.23)$$

Eq. (5.22) then defines a system of n algebraic equation to be solved for the n unknowns $c_1, \ldots c_n$. Thus Galerkin's method is applicable in the solution of variational problems. However, much of its value lies in the fact that it is not necessarily connected with a variational procedure.

B. EIGENVALUE PROBLEMS

Suppose that it is desired to solve an eigenvalue problem of the form

$$LU = \lambda U(x,y) \qquad \text{in } A, \qquad (5.24)$$

and

$$U = 0 \qquad \text{on } \delta A. \qquad (5.25)$$

Assume as in the first section that the function $U(x,y)$ can be approximated in the form of Eq. (5.23). The problem is solved by considering equations of the form

$$\iint_A \left[L(U_n) - \lambda U_n\right] \Phi_j \, dxdy = 0 \qquad (5.26)$$

for $j = 1, 2, \ldots, n$.

Now by substituting for U_n the functions Φ_k a system is obtained of the form

$$\sum_{j=1}^{n} (\alpha_{i,j} - \lambda\gamma_{i,j})C_j = 0 \qquad (5.27)$$

$$\text{for } i = 1, 2, \ldots, n,$$

where

$$\alpha_{i,j} = \iint_A L(\Phi_i)\ \Phi_j\ dxdy \qquad (5.28)$$

$$\gamma_{i,j} = \iint_A \Phi_i \Psi_j\ dxdy \qquad (5.29)$$

This has nontrivial solutions for the C's if, and only if,

$$\Delta = |\alpha_{i,j} - \lambda\gamma_{i,j}| = 0, \qquad (5.30)$$

where Δ is the determinant of the matrix. Thus the values of λ_i may be found by solving the characteristic equation (5.30). For each eigenvalue λ_i there corresponds a system of equations (5.28) to be solved for the eigenvector

$$\sum_{k=1}^{n} (\alpha_{k,j} - \lambda_i\gamma_{k,j})\ C_k = 0 \qquad (5.31)$$

$$\text{for } j = 1, 2, \ldots, n.$$

Now for each eigenvalue λ_i this system of n homogeneous equations may be solved to give the values $C_k^{(i)}$, where this coefficient corresponding to the ith eigenvalue.

Thus the eigenvalues are obtained, with their corresponding eigenvectors. The functions Φ_i should be chosen to reflect whatever characteristics the solution is felt to have.

VI. DISCUSSION OF VARIOUS METHODS FOR SOLVING EIGENVALUE PROBLEMS

In Chapters II, III and IV, three methods were developed for finding the first three eigenvalues and eigenfunctions. All three gave satisfactory values for the eigenvalues and eigenfunctions, and satisfactory times for the rather simple problems considered here. These methods were used to obtain solutions of the following two problems:

$$U_{xx} + U_{yy} = - \lambda^2 U \qquad \text{in } A, \qquad (6.1)$$

subject to the constraint

$$U(x,y) = 0 \qquad \text{on } \delta A; \qquad (6.2)$$

and

$$\nabla^4 U = \lambda^2 U \qquad \text{in } A, \qquad (6.3)$$

subject to the constraints

$$U(x,y) = \frac{\partial^2 U}{\partial n^2} = 0 \qquad \text{on } \delta A. \qquad (6.4)$$

In this chapter the computation and numerical results are discussed and compared.

In all of the methods a set of functions was needed. In the Rayleigh-Ritz method these were the basis for the approximating functions; in the other two methods they were the initial estimates of the functions. The ones usually chosen were

$$U_1^0 = (x - x^2)(y - y^2), \tag{6.5}$$

$$U_2^0 = (x - x^2)(y - y^2)(\tfrac{1}{2} - x), \tag{6.6}$$

$$U_3^0 = (x - x^2)(y - y^2)(\tfrac{1}{2} - y). \tag{6.7}$$

Step functions were also used as first estimates in the iterative methods; these increased the number of iterations required some, but not much, particularly for the higher modes.

Usually the range of the independent variable was divided into ten equal sub-intervals, in going to a difference equation. In the Rayleigh-Ritz method this did not yield sufficient accuracy and it was found necessary to go to forty intervals. Twenty-five intervals were also used in some computations; the intermediate number was chosen because of storage requirements in dynamic programming.

In the iterative procedures some convergence or stop criterion was needed. When a relaxation procedure was used together with Rayleigh's formula for estimating the eigenvalue, computation was terminated whenever the eigenvalue did not decrease by at least 0.002. In Stodola's method, the routine was terminated whenever the norm of the change in the function $U(x,y)$ was less than 0.002.

The relaxation method required the least time for this simple problem. Most of the time required for dynamic programming was spent in inverting the matrices. Dynamic programming yielded a very accurate approximation to the eigenfunction.

For some reason it was necessary to use forty intervals to get the desired accuracy with the Rayleigh-Ritz method. When ten intervals were used, the eigenvalues of the matrix $\bar{E}$ were significantly too small, apparently due to errors in the integration and transformation routines. There were at least two other disadvantages of the Rayleigh-Ritz method. First it was tedious to program. Second, there may be some difficulties in choosing the functions Φ_1, particularly if some of the higher modes are desired. The results of a set of computations are given in Computer Output 5 , including the three eigenvalues, the associated coefficients and the values of the corresponding functions at various points. The routine is shown in Computer Program 5. The necessary matrix transformations and solutions for the eigenvalues were carried out using programs TRED2 and TGL2 respectively, [7].

The simple relaxation method of Chapter II had the advantage of being the simplest to program and to run. It had the disadvantages that terminal convergence was slower than in the method of dynamic programming and if a large number of points were involved the computing time increased greatly. The results of a set of computations are given in Computer Output 1. The routine is shown in Computer Program 1.

Dynamic programming had the advantage of yielding very accurate values in a small number of iterations. The disadvantages were that it was relatively difficult to program

and that it took quite a bit of time to generate the inverses of the matrices required. Computer Output 7 shows the results by this method and the program used is in Computer Program 6.

The values of the eigenvalues obtained by the three methods are compared in Table I. In Table I are the eigenvalues for the first differential equation, the difference equation with ten intervals, together with the results of the computations, the number in parenthesis by an entry indicates the number of intervals used in the computation. The eigenvalues for forty intervals are intermediate between those for ten and those for the differential equation.

Differential Equation	Difference Equation(10)	Rayleigh	Relaxation	Dynamic Programming
4.44289	4.42463	4.4110 (25)	4.42486(10)	4.42442(10)
		4.44751(40)	4.44611(40)	4.43753(25)
7.02482	6.92714	7.23029(40)	6.92736(10)	6.92717(10)
7.02482	6.92714	7.24707(40)	6.92736(10)	6.92717(10)

Table I.

Comparison of First Three Eigenvalues for Eq. (1.1)

The three methods gave close agreement for the first eigenvalue but tended to differ on the next two.

For the differential equation of higher order, only relaxation and dynamic programming were compared; these comparisons are shown in Table II using ten intervals.

Differential Equation	Difference Equation(10)	Relaxation	Dynamic Programming
19.739227	19.577361	19.60767	19.57777
49.348040	47.985220	48.00555	47.98473
49.348040	47.985220	48.02386	47.98474

Table II.

Comparison of First Three Eigenvalues for Eq. (1.3)

Computing times were similar, around twelve seconds for each. Computer Output 2 shows the results for the relocation method, and the program is Computer Program 2. Computer Output 9 shows the results for dynamic programming and the program is Computer Program 6.

VII. DISCUSSION OF THE SOLUTION OF A PARTIAL DIFFERENTIAL EQUATION BY VARIOUS METHODS

In Chapters III, IV, and V, three methods were developed for solving partial differential equations. These methods were used to obtain solutions of the following problems:

$$U_{xx} + U_{yy} = 2(x^2 + y^2 - x - y) \qquad \text{in A,} \qquad (7.1)$$

subject to the constraint

$$U(x,y) = 0 \qquad \text{on } \delta A; \qquad (7.2)$$

and

$$\nabla^4 U = 8 \qquad \text{in A,} \qquad (7.3)$$

subject to the constraints

$$U(x,y) = 0 \qquad \text{on } \delta A$$

$$\frac{\partial^2 U}{\partial n^2} = 2(y - y^2) \qquad \text{for } x = 0 \qquad (7.4)$$

$$\frac{\partial^2 U}{\partial n^2} = -2(y - y^2) \qquad \text{for } x = 1$$

$$\frac{\partial^2 U}{\partial n^2} = 2(x - x^2) \qquad \text{for } y = 0$$

$$\frac{\partial^2 U}{\partial n^2} = -2(x - x^2) \qquad \text{for } y = 0$$

In this chapter the computation and numerical results are discussed.

In Galerkin's method, an approximation for the function U(x,y) satisfying the constraint Eq. (7.2) was made by choosing suitable functions $\Phi_1(x,y)$, $\Phi_2(x,y)$ and $\Phi_3(x,y)$ to satisfy the constraint in Eq. (7.2). The approximation for U(x,y) was

$$U(x,y) = C_1\Phi_1(x,y) + C_2\Phi_2(x,y) + C_3\Phi_3(x,y) \quad (7.5)$$

$$= (x - x^2)(y - y^2)(C_1 + yxC_2 + x^2y^2C_3).$$

The values of C_1, C_2 and C_3 were obtained by techniques described in Chapter V. Computer Output 3 shows the values of C_1, C_2, C_3 and the function U(x,y) at various points. In this method the region A was sub-divided into ten equal sub-intervals for each independent variable for the integration routine. A second approximation for U(x,y) was made for this method as

$$U(x,y) = C_1\Phi_1 + C_2\Phi_2 + C_3\Phi_3 \quad (7.6)$$

$$= (x - x^2)(y - y^2)(xC_1 + yC_2 + xyC_3).$$

The purpose of this was as follows. There generally is some skill and art involved in choosing the functions $\Phi_1, \ldots, \Phi_n$ well. In fact in the first choice Φ_1 is actually the desired function. The second set of functions was chosen so as to get some feel for the consequences of a poor choice of the Φ_1's. The values of C_1, C_2, C_3 and the function at various points is shown in Computer Output 10.

The numerical solution of Eq. (7.1) by dynamic programming with the constraint of (7.2) yielded the values in

Computer Output 6. Again the region was divided as was done in Galerkin's method.

In the Rayleigh-Ritz method, an approximation for the solution, $U(x,y)$, satisfying Eq. (7.1) was made by choosing suitable functions $\Phi_1(x,y)$, $\Phi_2(x,y)$ and $\Phi_3(x,y)$ to satisfy the constraint (7.2). The approximation for $U(x,y)$ was

$$U(x,y) = C_1\Phi_1 + C_2\Phi_2 + C_3\Phi_3 \qquad (7.7)$$

$$= (x - x^2)(y - y^2)(C_1 + xyC_2 + x^2y^2C_3).$$

The values of C_1, C_2, and C_3 were obtained by techniques described in Chapter IV. Computer Output 4 shows the values of C_1, C_2, C_3 and the values obtained for $U(x,y)$ at various points. In this method it was found necessary to sub-divide the region into forty equal sub-intervals for each independent variable in order tc get satisfactory accuracy.

The best approximation to $U(x,y)$ was obtained by Galerkin's method using Eq. (7.5), where Φ_1 was the desired function. There was no error and the method was able to detect that this was the case. However, when Eq. (7.6) was used the maximum error was eight thousandths. The time required to solve the problem by this method was 0.57 seconds.

The problem was solved by dynamic programming in three seconds with accuracy to six digits. However, over half of the computer time was spent obtaining inverses, which could have been fed as data from solving Eq. (6.1) in this particular case.

The Rayleigh-Ritz method required just under ten seconds and had a maximum error of two thousandths.

Because of the time required and the accuracy obtained by Rayleigh-Ritz, only Galerkin's method and dynamic programming were used to solve Eq. (7.3).

Galerkin's method obtained the same values and accuracy in the solution of Eq. (7.3) as it did in the solution to Eq. (7.1) and took the same time.

Dynamic programming obtained the same degree of accuracy in solving Eq. (7.3) as it did in solving Eq. (7.1) and took three seconds. The results are shown in Computer Output 8 and the program is Computer Program 7.

VIII. CONCLUSION

The three methods considered for eigenvalue problems yielded satisfactory values for the eigenvalues and the eigenfunctions. Generally, the relaxation method seemed to be most satisfactory. It was straight-forward to program, and it was faster then Rayleigh-Ritz and dynamic programming. It converged rapidly even if step functions were used on several different tests figures, such as the L-shaped and triangular regions.

The dynamic programming method converged in the same number of iterations as relaxation, but gave poorer estimates of the second and third eigenvalues. It of course was much more difficult to program and required more computing time due to the needed inverses.

The Rayleigh-Ritz method seemed to have little to recommend it due to the computer time required. It required much finer meshing in order to obtain a satisfactory accuracy. The only advantage it had was that no iteration was required.

Of the three methods considered in the solution of Poisson's and the biharmonic equations Galerkin's method was the fastest and gave accuracy comparable to the Rayleigh-Ritz method. It was also the simplest of the three methods used to program.

Dynamic programming gave the best accuracy generally, but it required more computer time than Galerkin's method.

Rayleigh-Ritz had the same difficulties as it did in the eigenvalue problem and was considered of little use.

While dynamic programming required more time for the solution of both eigenvalue problems and elliptic partial differential equations, it was very powerful. It only required a knowledge of the function U on the boundary. It can be extended to irregular regions [1]. It obtained very good accuracy. Much of the time was spent computing the inverses. If the same points were used in solving several different problems, these inverses could be calculated once and thereafter entered as data, reducing the computer time greatly.

COMPUTER OUTPUT 1.(RELAXATION)

EIGENFUNCTION=1

PATH= 8 OMEGA= 4.424858

X	Y	U(X,Y)
0.1	0.1	0.09719044
0.1	0.3	0.2508801
0.1	0.5	0.3094832
0.1	0.7	0.2526796
0.1	0.9	0.09705645
0.3	0.1	0.2508801
0.3	0.3	0.6512997
0.3	0.5	0.8061690
0.3	0.7	0.6586339
0.3	0.9	0.2531579
0.5	0.1	0.3094832
0.5	0.3	0.8061690
0.5	0.5	1.000000
0.5	0.7	0.8169372
0.5	0.9	0.3137754
0.7	0.1	0.2526796
0.7	0.3	0.6586339
0.7	0.5	0.8169372
0.7	0.7	0.6658412
0.7	0.9	0.2551157
0.9	0.1	0.09705645
0.9	0.3	0.2531579
0.9	0.5	0.3137754
0.9	0.7	0.2551157

COMPUTER OUTPUT 1.(RELAXATION)

EIGENFUNCTION=2

PATH=11 OMEGA= 6.927361

X	Y	U(X,Y)
0.1	0.1	0.1853049
0.1	0.3	0.4886459
0.1	0.5	0.6086538
0.1	0.7	0.4958898
0.1	0.9	0.1895596
0.3	0.1	0.3032387
0.3	0.3	0.8008109
0.3	0.5	0.9981067
0.3	0.7	0.8116993
0.3	0.9	0.3099311
0.5	0.1	0.001278793
0.5	0.3	0.008393560
0.5	0.5	0.01233941
0.5	0.7	0.008128799
0.5	0.9	0.001685064
0.7	0.1	-0.3083511
0.7	0.3	-0.8038315
0.7	0.5	-0.9953710
0.7	0.7	-0.8089499
0.7	0.9	-0.3103257
0.9	0.1	-0.1921543
0.9	0.3	-0.5006261
0.9	0.5	-0.6194299
0.9	0.7	-0.5025631

COMPUTER OUTPUT 1.(RELAXATION)

EIGENFUNCTION=3

PATH=11 OMEGA= 6.927361

X	Y	U(X,Y)
0.1	0.1	0.1857997
0.1	0.3	0.3045443
0.1	0.5	0.002898388
0.1	0.7	-0.3070407
0.1	0.9	-0.1916554
0.3	0.1	0.4894530
0.3	0.3	0.8029368
0.3	0.5	0.01102810
0.3	0.7	-0.8017028
0.3	0.9	-0.4998162
0.5	0.1	0.6086553
0.5	0.3	0.9981196
0.5	0.5	0.01235584
0.5	0.7	-0.9953650
0.5	0.9	-0.6194320
0.7	0.1	0.4950783
0.7	0.3	0.8095817
0.7	0.5	0.005508117
0.7	0.7	-0.8110784
0.7	0.9	-0.5033802
0.9	0.1	0.1890559
0.9	0.3	0.3086166
0.9	0.5	5.926900'-05
0.9	0.7	-0.3116446

COMPUTER OUTPUT 2.(RELAXATION)

EIGENFUNCTION=1

PATH=17 OMEGA= 19.60767

X	Y	U(X,Y)
0.1	0.1	0.1011795
0.1	0.3	0.2574610
0.1	0.5	0.3156426
0.1	0.7	0.2600489
0.1	0.9	0.1008759
0.3	0.1	0.2580318
0.3	0.3	0.6597930
0.3	0.5	0.8128256
0.3	0.7	0.6714125
0.3	0.9	0.2608653
0.5	0.1	0.3149233
0.5	0.3	0.8084638
0.5	0.5	1.000000
0.5	0.7	0.8271890
0.5	0.9	0.3214785
0.7	0.1	0.2583916
0.7	0.3	0.6639176
0.7	0.5	0.8214287
0.7	0.7	0.6778069
0.7	0.9	0.2627081
0.9	0.1	0.1004035
0.9	0.3	0.2579977
0.9	0.5	0.3188558

COMPUTER OUTPUT 2.(RELAXATION)

EIGENFUNCTION=2

PATH=16 OMEGA= 48.00555

X	Y	U(X,Y)
0.1	0.1	0.1909801
0.1	0.3	0.4904021
0.1	0.5	0.6059132
0.1	0.7	0.4979762
0.1	0.9	0.1925782
0.3	0.1	0.3037586
0.3	0.3	0.7884125
0.3	0.5	0.9826701
0.3	0.7	0.8096838
0.3	0.9	0.3130895
0.5	0.1	0.003429962
0.5	0.3	0.01207077
0.5	0.5	0.01897281
0.5	0.7	0.01404283
0.5	0.9	0.004551671
0.7	0.1	-0.3064085
0.7	0.3	-0.7943833
0.7	0.5	-0.9861545
0.7	0.7	-0.8114642
0.7	0.9	-0.3132687
0.9	0.1	-0.1937618
0.9	0.3	-0.5022484
0.9	0.5	-0.6227177

COMPUTER OUTPUT 2.(RELAXATION)

EIGENFUNCTION=3

PATH=19 OMEGA= 48.02386

X	Y	U(X,Y)
0.1	0.1	0.1851881
0.1	0.3	0.3020115
0.1	0.5	0.009329475
0.1	0.7	-0.3049613
0.1	0.9	-0.1935259
0.3	0.1	0.4791670
0.3	0.3	0.7890626
0.3	0.5	0.02798434
0.3	0.7	-0.7933792
0.3	0.9	-0.5036968
0.5	0.1	0.5902148
0.5	0.3	0.9785894
0.5	0.5	0.03599443
0.5	0.7	-0.9824019
0.5	0.9	-0.6229793
0.7	0.1	0.4810330
0.7	0.3	0.7979080
0.7	0.5	0.02484581
0.7	0.7	-0.8038256
0.7	0.9	-0.5073704
0.9	0.1	0.1851680
0.9	0.3	0.3065956
0.9	0.5	0.007537059

COMPUTER OUTPUT 3.(GALERKIN)

VALUES OF C(I) ARE

C1 = 1.000000
C2 = 0
C3 = 0

X	Y	U(X,Y)
0.1	0.1	0.008099992
0.1	0.3	0.01889999
0.1	0.5	0.02249999
0.1	0.7	0.01890001
0.1	0.9	0.008100022
0.3	0.1	0.01889999
0.3	0.3	0.04409999
0.3	0.5	0.05249999
0.3	0.7	0.04410003
0.3	0.9	0.01890005
0.5	0.1	0.02249999
0.5	0.3	0.05249999
0.5	0.5	0.06250000
0.5	0.7	0.05250004
0.5	0.9	0.02250007
0.7	0.1	0.01890001
0.7	0.3	0.04410003
0.7	0.5	0.05250004
0.7	0.7	0.04410006
0.7	0.9	0.01890007

COMPUTER OUTPUT 4.(RAYLEIGH)

VALUES OF C(I) ARE

C1 = 1.120410

C2 = -0.5808935

C3 = 0.3500372

X	Y	U(X,Y)
0 .25	0 .25	0.03816108
0 .25	0 .50	0.04937190
0 .25	0 .75	0.03599290
0 .50	0 .25	0.04937190
0 .50	0 .50	0.06231650
0 .50	0 .75	0.04461559
0 .75	0 .25	0.03599290
0 .75	0 .50	0.04461559
0 .75	0 .75	0.03179573

COMPUTER OUTPUT 5.(RAYLEIGH)

EIGENFUNCTION=1 OMEGA= 4.447512

THE VALUES OF C(I) ARE

C1 = 29.89044
C2 = 0.003808264
C3 = 2.580980

X		Y		U(X,Y)
0	.25	0	.25	1.073552
0	.25	0	.50	1.401157
0	.25	0	.75	1.028184
0	.50	0	.25	1.431359
0	.50	0	.50	1.868153
0	.50	0	.75	1.370868
0	.75	0	.25	1.073485
0	.75	0	.50	1.401070
0	.75	0	.75	1.028116

COMPUTER OUTPUT 5.(RAYLEIGH)

EIGENFUNCTION=2 OMEGA= 7.230292

THE VALUES OF C(I) ARE

C1 = 9.664003
C2 = 112.0784
C3 = -112.0294

X		Y		U(X,Y)
0	.25	0	.25	0.3401793
0	.25	0	.50	1.766417
0	.25	0	.75	2.309447
0	.50	0	.25	-0.8598440
0	.50	0	.50	0.6040002
0	.50	0	.75	1.765845
0	.75	0	.25	-1.629948
0	.75	0	.50	-0.8604184
0	.75	0	.75	0.3393202

COMPUTER OUTPUT 5.(RAYLEIGH)

EIGENFUNCTION=3 OMEGA= 7.247070

THE VALUES OF C(I) ARE

C1 = 9.656152
C2 = -112.4433
C3 = -111.6595

X		Y		U(X.Y)
0	.25	0	.25	-1.630179
0	.25	0	.50	-0.8650630
0	.25	0	.75	0.3325842
0	.50	0	.25	-0.8558773
0	.50	0	.50	0.6035087
0	.50	0	.75	1.761141
0	.75	0	.25	0.3463630
0	.75	0	.50	1.770327
0	.75	0	.75	2.309128

COMPUTER OUTPUT 6.(DYNAMIC)

X	Y	U(X,Y)
0.1	0.1	0.008099854
0.1	0.3	0.01889967
0.1	0.5	0.02249958
0.1	0.7	0.01889964
0.1	0.9	0.008099858
0.3	0.1	0.01889966
0.3	0.3	0.04409914
0.3	0.5	0.05249893
0.3	0.7	0.04409913
0.3	0.9	0.01889965
0.5	0.1	0.02249958
0.5	0.3	0.05249896
0.5	0.5	0.06249879
0.5	0.7	0.05249900
0.5	0.9	0.02249960
0.7	0.1	0.01889965
0.7	0.3	0.04409916
0.7	0.5	0.05249896
0.7	0.7	0.04409920
0.7	0.9	0.01889972
0.9	0.1	0.008099854
0.9	0.3	0.01889966
0.9	0.5	0.02249962
0.9	0.7	0.01889972
0.9	0.9	0.008099906

COMPUTER OUTPUT 7.(DYNAMIC)

EIGENFUNCTION=1

PATH= 4 OMEGA= 4.423473

X	Y	U(X,Y)
0.1	0.1	0.09554338
0.1	0.3	0.2500933
0.1	0.5	0.3091003
0.1	0.7	0.2500929
0.1	0.9	0.09554338
0.3	0.1	0.2500930
0.3	0.3	0.6546414
0.3	0.5	0.8090985
0.3	0.7	0.6546412
0.3	0.9	0.2500929
0.5	0.1	0.3091002
0.5	0.3	0.8090987
0.5	0.5	1.000000
0.5	0.7	0.8090994
0.5	0.9	0.3091004
0.7	0.1	0.2500930
0.7	0.3	0.6546419
0.7	0.5	0.8090992
0.7	0.7	0.6546427
0.7	0.9	0.2500940
0.9	0.1	0.09554315
0.9	0.3	0.2500930
0.9	0.5	0.3091010

COMPUTER OUTPUT 7.(DYNAMIC)

EIGENFUNCTION=2

PATH= 6 OMEGA= 6.927172

X	Y	U(X,Y)
0.1	0.1	0.1913275
0.1	0.3	0.5004815
0.1	0.5	0.6183150
0.1	0.7	0.5004799
0.1	0.9	0.1913269
0.3	0.1	0.3092604
0.3	0.3	0.8089828
0.3	0.5	0.9994567
0.3	0.7	0.8089805
0.3	0.9	0.3092589
0.5	0.1	1.959012'-06
0.5	0.3	3.750642'-06
0.5	0.5	2.407420'-06
0.5	0.7	-4.548319'-07
0.5	0.9	-7.790408'-07
0.7	0.1	-0.3092566
0.7	0.3	-0.8089775
0.7	0.5	-0.9994535
0.7	0.7	-0.8089827
0.7	0.9	-0.3092611
0.9	0.1	-0.1913257
0.9	0.3	-0.5004784
0.9	0.5	-0.6183146

COMPUTER OUTPUT 7.(DYNAMIC)

EIGENFUNCTION=3

PATH=11 OMEGA= 6.927173

X	Y	U(X,Y)
0.1	0.1	0.1913257
0.1	0.3	0.3092576
0.1	0.5	-1.199204'-06
0.1	0.7	-0.3092595
0.1	0.9	-0.1913273
0.3	0.1	0.5004785
0.3	0.3	0.8089781
0.3	0.5	-1.801975'-06
0.3	0.7	-0.8089815
0.3	0.9	-0.5004808
0.5	0.1	0.6183135
0.5	0.3	0.9994553
0.5	0.5	1.738089'-06
0.5	0.7	-0.9994555
0.5	0.9	-0.6183147
0.7	0.1	0.5004803
0.7	0.3	0.8089837
0.7	0.5	4.765801'-06
0.7	0.7	-0.8089792
0.7	0.9	-0.5004812
0.9	0.1	0.1913269
0.9	0.3	0.3092602
0.9	0.5	2.622819'-06

COMPUTER OUTPUT 8.(DYNAMIC)

X	Y	U(X,Y)
0.1	0.1	0.008099731
0.1	0.3	0.01889934
0.1	0.5	0.02249917
0.1	0.7	0.01889930
0.1	0.9	0.008099727
0.3	0.1	0.01889932
0.3	0.3	0.04409828
0.3	0.5	0.05249788
0.3	0.7	0.04409826
0.3	0.9	0.01889930
0.5	0.1	0.02249917
0.5	0.3	0.05249792
0.5	0.5	0.06249751
0.5	0.7	0.05249795
0.5	0.9	0.02249918
0.7	0.1	0.01889931
0.7	0.3	0.04409831
0.7	0.5	0.05249793
0.7	0.7	0.04409834
0.7	0.9	0.01889938
0.9	0.1	0.008099716
0.9	0.3	0.01889932
0.9	0.5	0.02249920
0.9	0.7	0.01889938
0.9	0.9	0.008099772

COMPUTER OUTPUT 9.(DYNAMIC)

EIGENFUNCTION=1

PATH= 3 OMEGA= 19.5763°

X	Y	U(X,Y)
0.1	0.1	0.09549332
0.1	0.3	0.2500034
0.1	0.5	0.3090197
0.1	0.7	0.2500030
0.1	0.9	0.09549332
0.3	0.1	0.2500032
0.3	0.3	0.6545131
0.3	0.5	0.8090189
0.3	0.7	0.6545127
0.3	0.9	0.2500032
0.5	0.1	0.3090197
0.5	0.3	0.8090194
0.5	0.5	1.000000
0.5	0.7	0.8090203
0.5	0.9	0.3090200
0.7	0.1	0.2500033
0.7	0.3	0.6545139
0.7	0.5	0.8090203
0.7	0.7	0.6545147
0.7	0.9	0.2500044
0.9	0.1	0.09549326
0.9	0.3	0.2500034
0.9	0.5	0.3090206

COMPUTER OUTPUT 9.(DYNAMIC)

EIGENFUNCTION=2

PATH= 4 OMEGA= 47.98485

X	Y	U(X,Y)
0.1	0.1	C.1911072
0.1	0.3	0.5001713
0.1	0.5	0.6181207
0.1	0.7	C.5001690
0.1	0.9	0.1911063
0.3	0.1	0.3091234
0.3	0.3	0.8090394
0.3	0.5	0.9998273
0.3	0.7	0.8090362
0.3	0.9	0.3091207
0.5	0.1	-9.384042'-07
0.5	0.3	8.968278'-06
0.5	0.5	1.008165'-05
0.5	0.7	3.054505'-06
0.5	0.9	-4.620023'-06
0.7	0.1	-C.3091281
0.7	0.3	-0.8090287
0.7	0.5	-0.9998166
0.7	0.7	-C.8090367
0.7	0.9	-0.3091323
0.9	0.1	-0.1911175
0.9	0.3	-0.5001757
0.9	0.5	-0.6181255

COMPUTER OUTPUT 9.(DYNAMIC)

EIGENFUNCTION=3

PATH= 7 OMEGA= 47.98486

X	Y	U(X,Y)
0.1	0.1	0.1911126
0.1	0.3	0.3091245
0.1	0.5	-9.687110'-07
0.1	0.7	-0.3091258
0.1	0.9	-0.1911120
0.3	0.1	0.5001729
0.3	0.3	0.8090308
0.3	0.5	-2.217028'-06
0.3	0.7	-0.8090366
0.3	0.9	-0.5001741
0.5	0.1	0.6181221
0.5	0.3	0.9998204
0.5	0.5	1.343255'-06
0.5	0.7	-0.9998225
0.5	0.9	-0.6181232
0.7	0.1	0.5001751
0.7	0.3	0.8090382
0.7	0.5	5.371387'-06
0.7	0.7	-0.8090329
0.7	0.9	-0.5001737
0.9	0.1	0.1911140
0.9	0.3	0.3091279
0.9	0.5	3.674680'-06

COMPUTER OUTPUT 10. (GALERKIN)

VALUES OF C(I) ARE

C1 = 1.596382
C2 = 1.596392
C3 = -2.598899

X	Y	U(X,Y)
0.1	0.1	0.002375632
0.1	0.3	0.01059511
0.1	0.5	0.01862749
0.1	0.7	0.02069907
0.1	0.9	0.01103618
0.3	0.1	0.01059507
0.3	0.3	0.03192532
0.3	0.5	0.04658196
0.3	0.7	0.04633235
0.3	0.9	0.02294400
0.5	0.1	0.01862741
0.5	0.3	0.04658184
0.5	0.5	0.05916639
0.5	0.7	0.05281764
0.5	0.9	0.02397245
0.7	0.1	0.02069896
0.7	0.3	0.04633217
0.7	0.5	0.05281758
0.7	0.7	0.04240138
0.7	0.9	0.01732974

COMPUTER PROGRAM 1.

RELAXATION METHOD APPLIED TO

$\nabla^2 U = -\lambda^2 U$, EQ. (1.1)

```
BEGIN
COMMENT FINDS EIGENVALUES AND EIGENVECTORS BY
 RELAXATION BASED ON RAYLEIGH'S FORMULA ON ANY
 TYPE OF DOMAIN.
   FORMAL PARAMETERS:
   OM       OMEGA
   OM2      OMEGA SQUARED
   OMO2     LAST VALUE OF OMEGA SQUARED
   ZNORM    MAX. NORM
   N1       MAX. NUMBER OF X POSITIONS
   P1       MAX. NUMBER OF Y POSITIONS
   H1       MESH OF X VARIABLE
   H2       MESH OF Y VARIABLE
   N2       ARRAY OF LOWER POSITIONS OF Y
   N3       ARRAY OF UPPER POSITIONS OF Y
   U,U1,Z   UNKNOWN EIGENFUNCTIONS
   PE       POTENTIAL ENERGY
   KE       KINETIC ENERGY;
REAL KE,PE,OM2,OMO2,ERR,ZS,OM,ZNORM;
REAL H1,H2,C1,C2,C3,CAT,DOG,CAT1,DOG1;
INTEGER N,M,L,N1,P1;
REAL H,H3,H4;
OM2:=5000.0;
N:=1;
ERR:=0.002;
L:=1;
CAT1:=5000.0;
READ(N1,P1,H1,H2);
H:=H1*H2;
H3:=H1**2;
H4:=H2**2;
BEGIN
REAL ARRAY X(0::N1);
REAL ARRAY Y(0::P1);
INTEGER ARRAY N2,N3(0::N1);
REAL ARRAY Z,U,U1(0::N1,0::P1);
FOR I:=0 UNTIL N1 DO
FOR J:=0 UNTIL P1 DO
   BEGIN U1(I,J):=0.0;
   Z(I,J):=0.0;
   END;
FOR I:=0 UNTIL N1 DO X(I):=I*H1;
FOR I:=0 UNTIL P1 DO Y(I):=I*H2;
FOR I:=0 UNTIL N1 DO READON(N2(I));
FOR I:=0 UNTIL N1 DO READON(N3(I));
9:
IF L=1 THEN
   BEGIN L:=2;
   FOR I:=0 UNTIL N1 DO
   FOR J:=0 UNTIL P1 DO
   Z(I,J):=(X(I)-X(I)**2)*(Y(J)-Y(J)**2);
   GO TO 9;
   END;
IF L=2 THEN
   BEGIN
   FOR I:=0 UNTIL N1 DO
   FOR J:=0 UNTIL P1 DO
      BEGIN
      U(I,J):=Z(I,J);
      Z(I,J):=(X(I)-X(I)**2)*(Y(J)-Y(J)**2)*(0.5-X(I));
      END;
   CAT:=KE;L:=3;N:=1;OM2:=5000.0;
```

```
    GO TO D;
    END;
IF L=3 THEN
    BEGIN
    FOR I:=0 UNTIL N1 DO
    FOR J:=0 UNTIL P1 DO
        BEGIN
        U1(I,J):=Z(I,J);
        Z(I,J):=(X(I)-X(I)**2)*(Y(J)-Y(J)**2)*(0.5-Y(J));
        END;
    CAT1:=KE;L:=4;N:=1;OM2:=5000.0;
    GO TO D;
    END;
COMMENT GET PE AND KE;
A:PE:=KE:=0.0;
FOR I:=0 UNTIL N1-1 DO
FOR J:=N2(I) UNTIL N3(I)-1 DO
    BEGIN
    PE:=PE+(((Z(I+1,J)-Z(I,J))/H1)**2)*H
      +(((Z(I,J+1)-Z(I,J))/H2)**2)*H;
   KE:=KE+(Z(I,J)**2)*H1*H2;
    END;
OMO2:=OM2;
OM2:=PE/KE;OM:=SQRT(OM2);
IF (OM2>OMO2) OR (N>40) THEN
    BEGIN
    WRITE("NOT CONVERGING");
    WRITE("N=");WRITEON(N);
    WRITE("OMO2 AND OM2 ARE");WRITEON(OMO2,OM2);
    GO TO R;
    END;
IF ABS(OM2-OMO2)<ERA THEN
BEGIN
IOCONTROL(3);
WRITE(" ");WRITE(" ");WRITE(" ");
WRITE("                              COMPUTER OUTPUT 1.(RELAXATION)
    ");
WRITE(" ");WRITE(" ");
INTRINSIZE:=1;
WRITE("                    EIGENFUNCTION=",L-1);WRITE(" ");
INTRINSIZE:=2;
WRITE("                    PATH=",N,"     OMEGA=",OM);
WRITE(" ");WRITE(" ");
WRITE("                          X                    Y      ",
    "      U(X,Y)");
WRITE(" ");
FOR I:=1 STEP 2 UNTIL N1-1 DO
FOR J:=1 STEP 2 UNTIL P1-1 DO
    BEGIN INTRINSIZE:=2;
    WRITE("                    ",I DIV N1,".",I REM N1,"      ",
      J DIV P1,".",J REM P1,Z(I,J));
    WRITE(" ");
END;END;
IF (ABS(OM2-OMO2)<ERA) AND (L<4) THEN GO TO B;
IF ABS(OM2-OMO2)<ERA THEN GO TO R;
N:=N+1;
COMMENT GET NEW Z;
FOR I:=1 UNTIL N1-1 DO
    BEGIN
    FOR J:=N2(I)+1 UNTIL N3(I)-1 DO
    Z(I,J):=(H4*(Z(I+1,J)+Z(I-1,J))+H3*(Z(I,J+1)+Z(I,J-1)))
      /(2.*H3+2.*H4-OM2*(H**2));
    END;
IF L>2 THEN
    BEGIN
    C1:=DOG:=DOG1:=0.0;
    FOR I:=1 UNTIL N1-1 DO
    FOR J:=N2(I)+1 UNTIL N3(I)-1 DO
        BEGIN
        C1:=C1+Z(I,J)*H;
        DOG:=DOG+Z(I,J)*U(I,J)*H;
        DOG1:=DOG1+Z(I,J)*U1(I,J)*H;
```

```
        END;
    C2:=DDG/CAT;C3:=DDG1/CAT1;
    FOR I:=1 UNTIL N1-1 DO
    FOR J:=N2(I)+1 UNTIL N3(I)-1 DO
      Z(I,J):=Z(I,J)-C1-C2*U(I,J)-C3*U1(I,J);
    END;
D:ZNORM:=0.0;
FOR I:=0 UNTIL N1 DO
FOR J:=N2(I) UNTIL N3(I) DO
    BEGIN
    ZS:=ABS(Z(I,J));
    IF ZS>ZNORM THEN ZNORM:=ZS;
    END;
FOR I:=0 UNTIL N1 DO
FOR J:=N2(I) UNTIL N3(I) DO Z(I,J):=Z(I,J)/ZNORM;
GO TO A;
END;
R:END.
```

COMPUTER PROGRAM 2.

RELAXATION METHOD APPLIED TO

$\nabla^4 U = \lambda^2 U$, EQ. (1.3)

```
BEGIN
COMMENT FINDS EIGENVALUES AND EIGENVECTORS BY
 RELAXATION BASED ON RAYLEIGH'S FORMULA FOR
 D  4(U)=W**2 U.
    FORMAL PARAMETERS:
    OM        OMEGA
    OM2       OMEGA SQUARED
    OMO2      LAST VALUE OF OMEGA SQUARED
    ZNORM     MAX. NORM
    H         MESH SIZE OF X AND Y
    U,U1,Z    UNKNOWN EIGENFUNCTIONS
    PE        POTENTIAL ENERGY
    KE        KINETIC ENERGY;
REAL ARRAY Z,U,U1(0::14,0::14);
REAL KE,PE,OM2,OMO2,ERA,ZS,OM,ZNORM,H,CAT,DOG,CAT1,DOG1;
REAL C1,C2,C3;
INTEGER N,L;
REAL ARRAY Y(0::10);
REAL ARRAY X(0::10);
LOGICAL SWITCH;
REAL OMO;
OM:=7000.0;
SWITCH:=TRUE;
CAT:=CAT1:=2000.0;
H:=0.1;OM2:=2000.0;N:=1;ERA:=0.003;L:=1;
OM2:=700000.0;
ERA:=0.005;
FOR I:=0 UNTIL 10 DO X(I):=Y(I):=I*H;
FOR I:=0 UNTIL 14 DO
FOR J:=0 UNTIL 14 DO Z(I,J):=U(I,J):=U1(I,J):=0.0;
BEGIN
PROCEDURE NORMA;
    BEGIN ZNORM:=0.0;
FOR I:=3 UNTIL 11 DO
FOR J:=3 UNTIL 11 DO
    BEGIN ZS:=ABS(Z(I,J));
    IF ZS>ZNORM THEN ZNORM:=ZS;END;
FOR I:=0 UNTIL 14 DO
FOR J:=0 UNTIL 14 DO Z(I,J):=Z(I,J)/ZNORM;
GO TO A;
    END NORMA;
PROCEDURE FUNCT;
BEGIN
IF L=1 THEN
    BEGIN L:=2;
    FOR I:=0 UNTIL 10 DO
    FOR J:=0 UNTIL 10 DO
      Z(I+2,J+2):=(X(I)-X(I)**2)*(Y(J)-Y(J)**2);
    NORMA;END;
IF L=2 THEN
    BEGIN L:=3;CAT:=KE;N:=1;OM2:=2000.0;
    OM:=7000.0;
    FOR I:=0 UNTIL 14 DO FOR J:=0 UNTIL 14 DO U(I,J):=Z(I,J);
    FOR I:=0 UNTIL 14 DO FOR J:=0 UNTIL 14 DO Z(I,J):=0.0;
    FOR I:=0 UNTIL 10 DO
    FOR J:=0 UNTIL 10 DO
      Z(I+2,J+2):=(X(I)-X(I)**2)*(Y(J)-Y(J)**2)*(.5-X(I));
    NORMA;END;
IF L=3 THEN
    BEGIN L:=4;CAT1:=KE;N:=1;OM2:=2000.0;
OM:=7000.0;
    FOR I:=0 UNTIL 14 DO FOR J:=0 UNTIL 14 DO U1(I,J):=Z(I,J)
```

```
    FOR I:=0 UNTIL 14 DO FOR J:=0 UNTIL 14 DO Z(I,J):=0.0;
    FOR I:=0 UNTIL 10 DO
    FOR J:=0 UNTIL 10 DO
      Z(I+2,J+2):=(X(I)-X(I)**2)*(Y(J)-Y(J)**2)*(.5-Y(J));
    NORMA;END;
    END FUNCT;
IF SWITCH THEN BEGIN SWITCH:=FALSE;FUNCT;END;
COMMENT GET PE AND KE;
A:PE:=KE:=0.0;
FOR I:=2 UNTIL 12 DO
FOR J:=2 UNTIL 12 DO
    BEGIN PE:=PE+(Z(I+1,J)-4.*Z(I,J)+Z(I-1,J)+Z(I,J+1)
      +Z(I,J-1))**2/(H*H);
    KE:=KE+(Z(I,J)*H)**2;END;
OMO:=OM;
OM2:=PE/KE;OM:=SQRT(OM2);
IF (OM>OMO) OR (N>60) THEN
    BEGIN WRITE("NOT CONVERGING ");
    WRITE(" ");WRITE("N,ZNORM,OMO,OM");
    WRITE(N,ZNORM,OMO,OM);WRITE(" ");
    FOR I:=1 UNTIL 11 DO
    WRITE(Z(3,I),Z(5,I),Z(7,I),Z(9,I),Z(11,I));
    GO TO R;END;
IF ABS(OMO-OM)<ERR THEN
    BEGIN IOCONTROL(3);
    WRITE(" ");WRITE(" ");WRITE(" ");
    WRITE("                          COMPUTER OUTPUT 2.",
      "(RELAXATION)");
    WRITE(" ");WRITE(" ");
    INTFIELDSIZE:=1;
    WRITE("                    EIGENFUNCTION=",L-1);WRITE(" ");
    INTFIELDSIZE:=2;
    WRITE("                    PATH=",N,"      OMEGA=",OM);
    WRITE(" ");WRITE(" ");
    WRITE("                                X                    Y      ",
      "    U(X,Y)");WRITE(" ");WRITE(" ");
    FOR I:=1 STEP 2 UNTIL 9 DO
    FOR J:=1 STEP 2 UNTIL 9 DO
    BEGIN INTFIELDSIZE:=2;
        WRITE("                ",I DIV 10,".",I REM 10,
          "      ",J DIV 10,".",J REM 10,Z(I+2,J+2));
        WRITE(" ");
        END;
    IF L<4 THEN FUNCT ELSE GO TO R;
    END;
N:=N+1;
COMMENT GET NEW Z;
FOR I:=3 UNTIL 11 DO
FOR J:=3 UNTIL 11 DO
    Z(I,J):=(8.*Z(I+1,J)+8.*Z(I-1,J)+8.*Z(I,J+1)
      +8.*Z(I,J-1)-Z(I+2,J)-Z(I,J-2)-2.*Z(I+1,J+1)
      -2.*Z(I-1,J-1)-2.*Z(I-1,J+1)-2.*Z(I+1,J-1)
      -Z(I-2,J)-Z(I,J+2))/(20.-OM2*H*H*H*H);
FOR I:=2 UNTIL 12 DO
    BEGIN Z(0,I):=-Z(4,I);Z(14,I):=-Z(10,I);
    Z(1,I):=-Z(3,I);Z(13,I):=-Z(11,I);
    Z(I,0):=-Z(I,4);Z(I,14):=-Z(I,10);
    Z(I,1):=-Z(I,3);Z(I,13):=-Z(I,11);END;
IF L>2 THEN
    BEGIN C1:=DOG:=DOG1:=0.0;
    FOR I:=3 UNTIL 11 DO
    FOR J:=3 UNTIL 11 DO
        BEGIN C1:=C1+Z(I,J)*H*H;
        DOG:=DOG+Z(I,J)*U(I,J)*H*H;
        DOG1:=DOG1+Z(I,J)*U1(I,J)*H*H;END;
    C2:=DOG/CAT;C3:=DOG1/CAT1;
    FOR I:=3 UNTIL 11 DO
    FOR J:=3 UNTIL 11 DO
      Z(I,J):=Z(I,J)-C1-C2*U(I,J)-C3*U1(I,J);
    FOR I:=2 UNTIL 12 DO
        BEGIN
        Z(0,I):=-Z(4,I);Z(14,I):=-Z(10,I);Z(1,I):=-Z(3,I);
```

```
        Z(13,I):=-Z(11,I);Z(I,0):=-Z(I,4);Z(I,14):=-Z(I,10);
        Z(I,1):=-Z(I,3);Z(I,13):=-Z(I,11);END;
    END;
IF L<5 THEN NORMA;
END;
R:END.
```

COMPUTER PROGRAM 3.

GALERKIN'S METHOD APPLIED TO

$\nabla^2 U = 2(x^2+y^2-x-y)$, EQ. (1.5)

```
BEGIN
COMMENT SOLVES PARTIAL DIFFERENTIAL EQUATIONS OF
 THE FORM L(U)=F BY GALERKIN'S METHOD ON ANY DOMAIN.
    FORMAL PARAMETERS:
    N       NUMBER OF EQUATIONS
    M2      MAX. NUMBER OF X POSITIONS
    M3      MAX. NUMBER OF Y POSITIONS
    N2      ARRAY OF LOWER Y POSITIONS
    N3      ARRAY OF UPPER Y POSITIONS
    F       KNOWN FUNCTION
    U       UNKNOWN FUNCTION
    C(I)    ESTIMATING FUNCTIONS
    D1(I)   L(ESTIMATING FUNCTIONS)
    A(I)    COEF. OF ESTIMATING FUNCTIONS;
REAL H,E,DOG,CAT,DEV,AMUL,T,DOM;
REAL H1;
REAL H2,H3;
INTEGER M2,M3;
INTEGER N,M,R;
E:=0.000001;R:=0;DEV:=1.0;
READ(N,M2,M3,H2,H3);
H1:=H2*H3;
BEGIN
REAL ARRAY X(0::M2);
REAL ARRAY Y(0::M3);
REAL ARRAY B(1::N,1::N+1);
REAL ARRAY A,C(1::N);
REAL ARRAY F,U(0::M2,0::M3);
REAL ARRAY D,D1(1::N,0::M2,0::M3);
INTEGER ARRAY N2,N3(0::M2);
FOR I:=0 UNTIL M2 DO X(I):=I*H2;
FOR I:=0 UNTIL M3 DO Y(I):=I*H3;
FOR I:=0 UNTIL M2 DO READON(N2(I));
FOR I:=0 UNTIL M2 DO READON(N3(I));
FOR I:=0 UNTIL M2 DO
FOR J:=0 UNTIL M3 DO
    BEGIN
    F(I,J):=0.0;
    U(I,J):=0.0;
    FOR L:=1 UNTIL N DO
      D(L,I,J):=D1(L,I,J):=0.0;
    END;
BEGIN COMMENT SOLVE HERE;
PROCEDURE WRITER;
BEGIN WRITE("SINGULAR");GO TO Q;
END WRITER;
PROCEDURE PERFORM;
BEGIN COMMENT GET U(I,J) AND A(I);
WRITE(" ");WRITE(" ");WRITE(" ");
WRITE("                              COMPUTER OUTPUT 3.(GALERKIN)");
WRITE(" ");WRITE(" ");
WRITE("                         VALUES OF C(I) ARE");WRITE(" ");
WRITE(" ");
FOR I:=1 UNTIL N DO
    BEGIN
    A(I):=B(I,N+1);
    INTFIELDSIZE:=1;
    WRITE("                              C",I,"=",A(I));WRITE(" ");
    END;
FOR J:=0 UNTIL M3-1 DO
BEGIN
FOR I:=0 UNTIL M2-1 DO
    BEGIN DOG:=0.0;
```

```
    FOR L:=1 UNTIL N DO DOG:=DOG+A(L)*D(L,I,J);
    U(I,J):=DOG;
END;END;
WRITE(" ");WRITE(" ");WRITE(" ");
WRITE("                    X                Y          ",
    "U(X,Y)");
WRITE(" ");WRITE(" ");
FOR I:=1 STEP 2 UNTIL M2-1 DO
FOR J:=1 STEP 2 UNTIL M3-1 DO
    BEGIN INTFIELDSIZE:=2;
    WRITE("                    ",I DIV M2,".",I REM M2,
    "      ",J DIV M3."."J REM M3,U(I,J));
    WRITE(" ");
    END;
GO TO Q;
END PERFORM;
PROCEDURE SWITCH;
BEGIN COMMENT SOLVES FOR A(I) COEFF;
INTEGER N1,M1;
M1:=N;N1:=N+1;
FOR I:=1 UNTIL M1 DO B(I,M1+1):=C(I);
FOR K:=1 UNTIL M1 DO
    BEGIN IF K=M1 THEN GO TO P;
    FOR I:=K+1 UNTIL M1 DO
        BEGIN IF ABS(B(K,K))<ABS(B(I,K)) THEN
            BEGIN R:=R+1;
            FOR J:=1 UNTIL N1 DO
                BEGIN T:=B(I,J);B(I,J):=B(K,J);B(K,J):=T;END;
        END;END;
    P:IF ABS(B(K,K))<E THEN WRITER ELSE
        BEGIN DEV:=DEV*B(K,K);DOM:=B(K,K);
        FOR J:=1 UNTIL N1 DO B(K,J):=B(K,J)/DOM;
        FOR I:=1 UNTIL M1 DO
            BEGIN AMUL:=B(I,K);
            IF I=K THEN  ELSE
                BEGIN
                FOR J:=1 UNTIL N1 DO
                    BEGIN B(I,J):=B(I,J)-AMUL*B(K,J);
    END;END;END;END;END;
PERFORM;
END SWITCH;
COMMENT GUESSED FUNCTIONS D(I) HERE;
FOR J:=0 UNTIL M2 DO
FOR K:=N2(J) UNTIL N3(J) DO
    BEGIN
    F(J,K):=2.*(X(J)**2+Y(K)**2-Y(K)-X(J));
    D(1,J,K):=(X(J)-X(J)**2)*(Y(K)-Y(K)**2);
    D(2,J,K):=(X(J)**2-X(J)**3)*(Y(K)**2-Y(K)**3);
    D1(1,J,K):=2.*(X(J)**2+Y(K)**2-X(J)-Y(K));
    D1(2,J,K):=(2.-6.*X(J))*(Y(K)**2-Y(K)**3)
        +(2.-6.*Y(K))*(X(J)**2-X(J)**3);
    END;
COMMENT SOLVE FOR INTEGRALS F*D AND LD*D;
FOR I:=1 UNTIL N DO
    BEGIN DOG:=0.0;
    FOR J:=1 UNTIL M2 DO
    FOR K:=N2(J)+1 UNTIL N3(J) DO
    DOG:=DOG+(F(J,K)*D(I,J,K)*H1);
    C(I):=DOG;
    FOR J:=1 UNTIL N DO
        BEGIN CAT:=0.0;
        FOR K:=1 UNTIL M2 DO
        FOR L:=N2(K)+1 UNTIL N3(K) DO
        CAT:=CAT+(D1(J,K,L)*D(I,K,L)*H1);
        B(I,J):=CAT;
        END;
    IF I=N THEN SWITCH;
    END;
END;END;
Q:END.
```

COMPUTER PROGRAM 4.

RAYLEIGH-RITZ METHOD APPLIED TO

$\nabla^2 U = 2(x^2+y^2-x-y)$, EQ. (1.1)

```
BEGIN
COMMENT SOLVES PARTIAL DIFFERENTIAL EQUATIONS BY
 RAYLEIGH-RITZ METHOD.
   FORMAL PARAMETERS:
    H2      DX SPACING
    H3      DY SPACING
    N       NUMBER OF EQUATIONS
    F       KNOWN FUNCTION
    M2      UPPER LIMIT OF X
    M3      UPPER LIMIT OF Y
    U       UNKNOWN FUNCTION
    N3      UPPER BOUNDS ON Y
    N2      LOWER BOUNDS ON Y
    A(I)    COEFF. OF ESTIMATING FUNCTIONS
    D(I)    ESTIMATING FUNCTIONS;
REAL H,E,DOG,CAT,DIV,AMUL,T,BOM,H1,H2,H3;
INTEGER M2,M3,N,M,E;
E:=0.000001;B:=0;DIV:=1.0;
READ(N,M2,M3,H2,H3);
H1:=H2*H3;
BEGIN
REAL ARRAY X(0::M2);
REAL ARRAY Y(0::M3);
REAL ARRAY B(1::N,1::N+1);
REAL ARRAY A,C(1::N);
REAL ARRAY F,U(0::M2,0::M3);
REAL ARRAY D,D1,D2(1::N,0::M2,0::M3);
INTEGER ARRAY N2,N3(0::M2);
FOR I:=0 UNTIL M2 DO X(I):=I*H2;
FOR I:=0 UNTIL M3 DO Y(I):=I*H3;
FOR I:=0 UNTIL M2 DO READ(N2(I));
FOR I:=0 UNTIL M2 DO READ(N3(I));
FOR I:=0 UNTIL M2 DO
FOR J:=0 UNTIL M3 DO
   BEGIN
   F(I,J):=U(I,J):=0.0;
   FOR L:=1 UNTIL N DO
   D(L,I,J):=D1(L,I,J):=D2(L,I,J):=0.0;
   END;
FOR I:=1 UNTIL N DO
FOR J:=1 UNTIL N+1 DO B(I,J):=0.0;
BEGIN
COMMENT SOLVE HERE;
PROCEDURE WRITER;
BEGIN WRITE("SINGULAR");GO TO Q;
END WRITER;
PROCEDURE PERFORM;
BEGIN
COMMENT GET U(I,J) AND A(I);
WRITE(" ");WRITE(" ");WRITE(" ");
WRITE("                    COMPUTER OUTPUT 4.(RAYLEIGH)");
WRITE(" ");WRITE(" ");WRITE(" ");
WRITE("                 VALUES OF C(I) ARE");WRITE(" ");
FOR I:=1 UNTIL N DO
   BEGIN
   A(I):=B(I,N+1);
   INTFIELDSIZE:=1;
   WRITE("                        C",I,"=",A(I));WRITE(" ");
   END;
FOR I:=1 UNTIL M2-1 DO
FOR J:=1 UNTIL M3-1 DO
   BEGIN DOG:=0.0;
   FOR L:=1 UNTIL N DO DOG:=DOG+A(L)*D(L,I,J);
```

```
    U(I,J):=DOG;
    END;
WRITE(" ");WRITE(" ");WRITE(" ");
WRITE("                    X                    Y      ",
    "      U(X,Y)");
WRITE(" ");WRITE(" ");
FOR I:=0 STEP 10 UNTIL M2 DO
FOR J:=0 STEP 10 UNTIL M3 DO
    BEGIN INTEGERSIZE:=2;
    WRITE("                    ",I DIV M2,".",I*100 REM M2,
    "      ",J DIV M2,".",J*100 REM M3,U(I,J));
    WRITE(" ");
    END;
GO TO Q;
END PERFORM;
PROCEDURE SWITCH;
BEGIN
COMMENT SOLVES FOR A(I) COEFF;
INTEGER N1,M1;
M1:=N;N1:=N+1;
FOR I:=1 UNTIL M1 DO B(I,M1+1):=C(I);
FOR K:=1 UNTIL M1 DO
    BEGIN
    IF K=M1 THEN GO TO P;
    FOR I:=K+1 UNTIL M1 DO
        BEGIN
        IF ABS(B(K,K))<ABS(B(I,K)) THEN
            BEGIN
            R:=R+1;
            FOR J:=1 UNTIL N1 DO
                BEGIN T:=B(I,J);B(I,J):=B(K,J);B(K,J):=T;END;
        END;END;
 P:IF ABS(B(K,K))<E THEN WRITE ELSE
        BEGIN
        DIV:=DIV*B(K,K);DOM:=B(K,K);
        FOR J:=1 UNTIL N1 DO B(K,J):=B(K,J)/DOM;
        FOR I:=1 UNTIL M1 DO
            BEGIN
            AMUL:=B(I,K);
            IF I=K THEN ELSE
                BEGIN
                FOR J:=1 UNTIL N1 DO
                    BEGIN
                    B(I,J):=B(I,J)-AMUL*B(K,J);
    END;END;END;END;END;
PERFORM;
END SWITCH;
COMMENT GUESSED FUNCTIONS O(I) HERE;
FOR J:=0 UNTIL M2 DO
FOR K:=N2(J) UNTIL N3(J) DO
    BEGIN
    F(J,K):=2.*(X(J)**2+Y(K)**2-X(J)-Y(K));
    O(1,J,K):=(X(J)-X(J)**2)*(Y(K)-Y(K)**2);
    O(2,J,K):=X(J)*Y(K)*O(1,J,K);
    O(3,J,K):=X(J)*Y(K)*O(2,J,K);
    O1(1,J,K):=(1.-2.*X(J))*(Y(K)-Y(K)**2);
    O1(2,J,K):=(2.*X(J)-3.*X(J)**2)*(Y(K)**2-Y(K)**3);
    O1(3,J,K):=(3.*X(J)**2-4.*X(J)**3)*(Y(K)**3-Y(K)**4);
    O2(1,J,K):=(1.-2.*Y(K))*(X(J)-X(J)**2);
    O2(2,J,K):=(2.*Y(K)-3.*Y(K)**2)*(X(J)**2-X(J)**3);
    O2(3,J,K):=(3.*Y(K)**2-4.*Y(K)**3)*(X(J)**3-X(J)**4);
    END;
COMMENT SOLVE FOR INTEGRALS NOW;
FOR I:=1 UNTIL N DO
    BEGIN
    DOG:=0.0;
    FOR J:=1 UNTIL M2 DO
    FOR K:=N2(J)+1 UNTIL N3(J) DO
    DOG:=DOG-(F(J,K)*O(I,J,K)*H1);
    C(I):=DOG;
    END;
FOR I:=1 UNTIL N DO
```

```
FOR J:=1 UNTIL N DO
    BEGIN
    FOR K:=1 UNTIL M2 DO
    FOR L:=N2(K)+1 UNTIL N3(K) DO
    B(I,J):=B(I,J)+(D1(I,K,L)*D1(J,K,L)+D2(I,K,L)*D2(J,K,L))*
      HI;
    END;
SWITCH;
END;END;
Q:END.
```

COMPUTER PROGRAM 5.

RAYLEIGH-RITZ METHOD APPLIED TO

$\nabla^2 U = -\lambda^2 U$, EQ. (1.1)

```
BEGIN
COMMENT SOLVES EIGENVALUE PROBLEMS BY RAYLEIGH-RITZ
 METHOD USING TRED2 AND TGL2 FROM NUMERISCHE
 MATHEMATIK 11,PP. 181-195 AND PP. 293-306(1968) BY MARTIN
 REINSCH,BOWDLER,HILARY,AND WILKINSON.
   FORMAL PARAMETERS:
   N      NUMBER OF EIGENVALUES
          AND THE NUMBER OF EQUATIONS USED
   HX     MESH OF X VARIABLE
   HY     MESH OF Y VARIABLE
   NPX    NUMBER OF X POSITIONS
   NPY    NUMBER OF Y POSITIONS
   U(I)   UNKNOWN EIGENFUNCTIONS
   C(I)   ESTIMATING FUNCTIONS USED
   R(I)   COEF. OF ESTIMATING FUNCTIONS
   D2(I)  EIGENVALUES OF MATRIX A
   D1(I)  EIGENVALUES OF MATRIX B
   Z1(I)  EIGENVECTORS OF MATRIX B
   Z2(I)  EIGENVECTORS OF MATRIX E=T'*A*T
   D(I)   USED IN TRANSFORMATION C=T'*D;
INTEGER NX,NY,N,NPX,NPY;
REAL HX,HY,EPS,DOG;
LOGICAL T;
PROCEDURE TRED2(INTEGER VALUE N;REAL ARRAY D,E(*);
   REAL ARRAY A,Z(*,*);REAL VALUE PU);
COMMENT REDUCES SYMMETRIC MATRIX TO TRIDIG. FORM USING
  HOUSEHOLDER,S REDUCTION.
  FORMAL PARAMETERS:
     N  ORDER OF SYMMETRIC MATRIX A
     D  DIAGONAL OF RESULTS
     E  SUB-DIAGONAL OF RESULTS;
   BEGIN
   INTEGER I,J,K,L;
   REAL F,G,H,HH,TOL;
   TOL:=PU/EPSILON;
   FOR I:=1 UNTIL N DO
   FOR J:=1 UNTIL I DO Z(I,J):=A(I,J);
   FOR I1:=N STEP -1 UNTIL 2 DO
       BEGIN I:=I1;
       L:=I-2;F:=Z(I,I-1);G:=0.0;
       FOR K:=1 UNTIL L DO G:=G+Z(I,K)*Z(I,K);
       H:=G+F*F;
       IF G <= TOL THEN
           BEGIN E(I):=F;H:=0.0;GO TO SKIP;END;
       L:=L+1;
       G:=E(I):=IF F>= 0.0 THEN -SQRT(H) ELSE SQRT(H);
       H:=H-F*G;Z(I,I-1):=F-G;F:=0.0;
       FOR J:=1 UNTIL L DO
           BEGIN
           Z(J,I):=Z(I,J)/H;G:=0.0;
           FOR K:=1 UNTIL J DO G:=G+Z(J,K)*Z(I,K);
           FOR K:=J+1 UNTIL L DO G:=G+Z(K,J)*Z(I,K);
           E(J):=G/H;F:=F+G*Z(J,I);
           END;
       HH:=F/(H+H);
       FOR J:=1 UNTIL L DO
           BEGIN
           F:=Z(I,J);G:=E(J):=E(J)-HH*F;
           FOR K:=1 UNTIL J DO
           Z(J,K):=Z(J,K)-F*E(K)-G*Z(I,K);
           END;
  SKIP:D(I):=H;
```

```
        END;
    D(1):=E(1):=0.0;
    FOR I:=1 UNTIL N DO
        BEGIN L:=I-1;
        IF D(I) ¬= 0.0 THEN
        FOR J:=1 UNTIL L DO
            BEGIN G:=0.0;
            FOR K:=1 UNTIL L DO G:=G+Z(I,K)*Z(K,J);
            FOR K:=1 UNTIL L DO Z(K,J):=Z(K,J)-G*Z(K,I);
            END;
        D(I):=Z(I,I);Z(I,I):=1.0;
        FOR J:=1 UNTIL L DO Z(I,J):=Z(J,I):=0.0;
        END;
    END TRED2;
PROCEDURE TQL2(INTEGER VALUE N;REAL ARRAY D,E(*);
    REAL ARRAY Z(*,*);LOGICAL RESULT FAIL);
COMMENT FINDS EIGENVALUES AND EIGENVECTORS.
  FORMAL PARAMETERS:
      N   ORDER TRIDIAGONAL MATRIX
      D   DIAGONAL ELEMENTS
      E   SUB-DIAGONAL ELEMENTS
      Z   MATRIX OF HOUSEHOLDER TRANSFORMATION;
    BEGIN
    INTEGER I,J,K,L,M;
    REAL B,C,F,G,H,P,R,S;
    FAIL:=FALSE;
    FOR I:=2 UNTIL N DO E(I-1):=E(I);
    E(N):=B:=F:=0.0;
    FOR L1:=1 UNTIL N DO
        BEGIN L:=L1;J:=0;
        H:=EPSILON*(ABS(D(L))+ABS(E(L)));
        IF B<H THEN B:=H;
        COMMENT LOOK FOR SMALL SUB-DIAGONAL ELEMENT;
        FOR M1:=L UNTIL N DO
            BEGIN M:=M1;
            IF ABS(E(M))<=B THEN GO TO CONT;END;
  CONT:IF M=L THEN GO TO ROOT;
NEXTIT:IF J=30 THEN
            BEGIN FAIL:=TRUE;GO TO FIN;END;
        J:=J+1;
        P:=(D(L+1)-D(L))/(2.*E(L));
        R:=SQRT(P**2+1.0);
        H:=D(L)-E(L)/(IF P<0.0 THEN P-R ELSE P+R);
        FOR I:=L UNTIL N DO D(I):=D(I)-H;
        F:=F+H;
        COMMENT QL TRANSFORMATION;
        P:=D(M):C:=1.0:S:=0.0;
        FOR I:=M-1 STEP -1 UNTIL L DO
            BEGIN
            G:=C*E(I):H:=C*P;
            IF ABS(P) >= ABS(E(I)) THEN
                BEGIN
                C:=E(I)/P:R:=SQRT(C**2+1.0);
                E(I+1):=S*P*R:S:=C/R:C:=1.0/R;
                END ELSE
                BEGIN
                C:=P/E(I):R:=SQRT(C**2+1.0);
                E(I+1):=S*E(I)*R:S:=1.0/R:C:=C/R;
                END;
            P:=C*D(I)-S*G;
            D(I+1):=H+S*(C*G+S*D(I));
            COMMENT FORM VECTOR;
            FOR K:=1 UNTIL N DO
                BEGIN
                H:=Z(K,I+1):Z(K,I+1):=S*Z(K,I)+C*H;
                Z(K,I):=C*Z(K,I)-S*H;
            END:END;
        E(L):=S*P:D(L):=C*P;
        IF ABS(E(L)) > B THEN GO TO NEXTIT;
  ROOT:D(L):=D(L)+F;
        END;
    COMMENT ORDER EIGENVALUES AND EIGENVECTORS;
```

```
    FOR I:=1 UNTIL N DO
        BEGIN K:=I;P:=D(I);
        FOR J:=I+1 UNTIL N DO
        IF D(J) < P THEN
            BEGIN K:=J;P:=D(J);END;
        IF K ¬= I THEN
            BEGIN
            D(K):=D(I);D(I):=P;
            FOR J:=1 UNTIL N DO
              BEGIN P:=Z(J,I);Z(J,I):=Z(J,K);Z(J,K):=P;END;
        END;END;
FIN:
    END TQL2;
START:READ(N,NPX,NPY,NX,NY,HX,HY,EPS);
BEGIN
COMMENT PROGRAM STARTS HERE;
REAL ARRAY D1,D2,E1,E2(1::N);
REAL ARRAY D,R(1::N,1::N);
REAL ARRAY U(1::N,0::NPX,0::NPY);
REAL ARRAY E,A,B,P(1::N,1::N);
REAL ARRAY X(0::NPX);
REAL ARRAY Y(0::NPY);
REAL ARRAY C,C1,C2(1::N,0::NPX,0::NPY);
REAL ARRAY Z1,Z2(1::N,1::N);
PROCEDURE FCN;
COMMENT COMPUTES FUNCTIONS AND MATRICES A AND B;
BEGIN
X(0):=Y(0):=0.0;
FOR I:=1 UNTIL NPX DO X(I):=I*HX;
FOR I:=1 UNTIL NPY DO Y(I):=I*HY;
FOR I:=1 UNTIL N DO
FOR J:=1 UNTIL N DO A(I,J):=B(I,J):=0.0;
FOR I:=0 UNTIL NPX DO
FOR J:=0 UNTIL NPY DO
    BEGIN
    C(1,I,J):=(X(I)-X(I)**2)*(Y(J)-Y(J)**2);
    C(2,I,J):=(0.5-X(I))*C(1,I,J);
    C(3,I,J):=(0.5-Y(J))*C(1,I,J);
    C1(1,I,J):=(1.-2.*X(I))*(Y(J)-Y(J)**2);
C1(2,I,J):=(Y(J)-Y(J)**2)*(0.5-3.*X(I)+3.*X(I)**2);
    C1(3,I,J):=(1.-2.*X(I))*(0.5-Y(J))*(Y(J)-Y(J)**2);
    C2(1,I,J):=(1.-2.*Y(J))*(X(I)-X(I)**2);
    C2(2,I,J):=(0.5-X(I))*(X(I)-X(I)**2)*(1.-2.*Y(J));
    C2(3,I,J):=(X(I)-X(I)**2)*(0.5-3.*Y(J)+3.*Y(J)**2);
    END;
FOR K:=1 UNTIL N DO
FOR L:=1 UNTIL N DO
    BEGIN
    FOR I:=1 UNTIL NPX DO
    FOR J:=1 UNTIL NPY DO
        BEGIN
        A(K,L):=A(K,L)+C1(K,I,J)*C1(L,I,J)*HX*HY
          +C2(K,I,J)*C2(L,I,J)*HX*HY;
        B(K,L):=B(K,L)+C(K,I,J)*C(L,I,J)*HX*HY;
        END;
    END;
END FCN;
PROCEDURE WRITER;
COMMENT WRITES ALL DATA;
BEGIN
FOR I:=1 UNTIL N DO
FOR J:=1 UNTIL I DO
    BEGIN DOG:=Z1(I,J);Z1(I,J):=Z1(J,I);Z1(J,I):=DOG;END;
FOR L:=1 UNTIL N DO
FOR I:=1 UNTIL N DO D(L,I):=Z2(I,L);
FOR L:=1 UNTIL N DO
FOR I:=1 UNTIL N DO
    BEGIN DOG:=0.0;
    FOR J:=1 UNTIL N DO DOG:=DOG+Z1(I,J)*D(L,J);
    R(I,L):=DOG;
    END;
FOR I:=1 UNTIL N DO
```

```
        BEGIN
        FOR J:=0 UNTIL NPX DO
        FOR L:=0 UNTIL NPY DO
            BEGIN U(I,J,L):=0.0;
            FOR K:=1 UNTIL N DO U(I,J,L):=U(I,J,L)+R(I,K)*
              C(K,J,L):
        END;END;
    FOR K:=1 UNTIL N DO
        BEGIN IOCONTROL(3);
        WRITE(" ");WRITE(" ");WRITE(" ");
        WRITE("                    COMPUTER OUTPUT 5.(RAYLEIGH)");
        WRITE(" ");WRITE(" ");WRITE(" ");
        WRITE("          EIGENFUNCTION=",K,"   OMEGA=",SQRT(D2(K)))
        WRITE(" ");WRITE(" ");WRITE(" ");
        WRITE("                THE VALUES OF C(I)");
        WRITE(" ");WRITE(" ");
        FOR L:=1 UNTIL N DO
            BEGIN INTFIELDSIZE:=1;
            WRITE("          C",L,"=",R(K,L));
            WRITE(" ");WRITE(" ");END;WRITE(" ");
        WRITE("          X                    Y                    U(X,Y)");
        WRITE(" ");WRITE(" ");
        FOR J:=0 STEP NX UNTIL NPX DO
        FOR L:=0 STEP NY UNTIL NPY DO
            BEGIN INTFIELDSIZE:=2;
            WRITE("          ",J DIV NPX,".",J REM NPX,"       ",
              L DIV NPY,".",L REM NPY,U(K,J,L));
            WRITE(" ");
        END;END;
        GO TO Q;
END WRITER;
FCN:
TRED2(N,D1,E1,B,Z1,EPS);
TQL2(N,D1,E1,Z1,T);
IF T THEN BEGIN WRITE("FAIL=1");GO TO Q;END ELSE
    BEGIN
    FOR I:=1 UNTIL N DO D1(I):=SQRT(D1(I));
    FOR I:=1 UNTIL N DO
    FOR J:=1 UNTIL N DO Z1(J,I):=Z1(J,I)/D1(I);
    END;
FOR I:=1 UNTIL N DO
FOR K:=1 UNTIL N DO
    BEGIN P(I,K):=0.0;
    FOR J:=1 UNTIL N DO P(I,K):=P(I,K)+A(I,J)*Z1(J,K);
    END;
FOR I:=1 UNTIL N DO
FOR J:=1 UNTIL I DO
    BEGIN DOG:=Z1(I,J):Z1(I,J):=Z1(J,I);Z1(J,I):=DOG;END;
FOR I:=1 UNTIL N DO
FOR K:=1 UNTIL N DO
    BEGIN E(I,K):=0.0;
    FOR J:=1 UNTIL N DO E(I,K):=E(I,K)+Z1(I,J)*P(J,K);
    END;
TRED2(N,D2,E2,E,Z2,EPS);
TQL2(N,D2,E2,Z2,T);
IF T THEN BEGIN WRITE("FAIL=2");GO TO Q;END;
WRITER;
END;
Q:GO TO START;
END.
```

COMPUTER PROGRAM 6.

DYNAMIC PROGRAMMING APPLIED TO

(1) $\nabla^2 U = 2(x^2+y^2-x-y)$ EQ. (1.5)

(2) $\nabla^4 U = \lambda^2 U$ EQ. (1.3)

```
BEGIN
COMMENT SOLVES PARTIAL DIFFERENTIAL EQUATIONS AND
 EIGENVALUE PROBLEMS ON A RECTANGLE BY DYNAMIC
 PROGRAMMING AND STODOLA'S METHOD.
   FORMAL  PARAMETERS:
   OM    OMEGA
   OM2   OMEGA SQUARED
   N     NUMBER OF X POSITIONS
   M     NUMBER OF Y POSITIONS
   K1    COEFF IF NEEDED
   V1    ORDER OF THE OPERATOR
           V1=0  FIRST ORDER EIGENVALUE PRB
           V1=2  SECOND ORDER EIGENVALUE PRB
   T1    TYPE OF PROGRAM RUN
           T1=0  FIRST ORDER
           T1=2  SECOND ORDER
           T1=50 EIGENVALUE
   P(I) SECOND NORMAL DERIVATIVE ON BDRY
   E1    FUNCTION TO SATISFY
   U     UNKNOWN FUNCTION
   Q1    NUMBER OF EIGENVALUES;
REAL KE,PE;
REAL E,ERA,OM,OM2,OMO2,H,K1;
REAL CAT1,CAT2,C1,C2,C3,DOG1,DOG2;
REAL ZNORM,ZS;
INTEGER N,M,N1,S,T1,V1,M1;
INTEGER Q1,Q2;
LOGICAL TOGGLE;
REAL ZNORM1;
X:READ(N,M,H,K1,T1,Q1,V1);
TOGGLE:=TRUE;
Q2:=1;
CAT1:=CAT2:=5000.0;
M1:=2*M-2;
OM2:=1000.0;
N1:=0;
ERA:=0.03;
ERA:=0.002;
E:=0.000001;
BEGIN
REAL ARRAY X1(0::N);
REAL ARRAY Y1(0::M);
REAL ARRAY U5(0::N,0::M);
REAL ARRAY U1,U2(0::N,0::M);
REAL ARRAY U3(0::N,0::M);
REAL ARRAY C1,B1,P1(0::N,0::M);
REAL ARRAY Z(0::N+4,0::M+4);
REAL ARRAY E1(0::N,0::M);
REAL ARRAY I1,Q(1::M-1,1::M-1);
REAL ARRAY U,B,R(0::N,0::M);
REAL ARRAY A(1::N,1::M,1::M);
REAL ARRAY D(1::N,0::M,0::M1);
REAL ARRAY CAT(0::M);
REAL ARRAY C(0::N,0::M);
REAL ARRAY P1,P3(0::M);
REAL ARRAY P2,P4(0::N);
FOR I:=0 UNTIL M DO READON(U(0,I));
FOR I:=0 UNTIL N DO READON(U(I,0));
FOR I:=0 UNTIL M DO READON(U(N,I));
FOR I:=0 UNTIL N DO READON(U(I,M));
FOR I:=0 UNTIL N DO X1(I):=I*H;
FOR J:=0 UNTIL M DO Y1(J):=J*H;
IF (T1>0) AND (V1=2) THEN
```

```
    BEGIN
    FOR I:=0 UNTIL M DO
    P1(I):=0.0;
    FOR I:=0 UNTIL M DO
    P3(I):=-P1(I);
    FOR I:=0 UNTIL N DO
    P2(I):=0.0;
    FOR I:=0 UNTIL N DO
    P4(I):=-P2(I);
    END;
FOR I:=0 UNTIL N DO
FOR J:=0 UNTIL M DO U1(I,J):=U2(I,J):=U3(I,J):=U5(I,J):=0.0;
S:=N;
BEGIN
PROCEDURE WRITER;
BEGIN WRITE("SINGULAR");
GO TO W;
END WRITER;
PROCEDURE WRIT1;
BEGIN
IOCONTROL(3);
WRITE(" ");WRITE(" ");WRITE(" ");
WRITE("                    COMPUTER OUTPUT 9.(DYNAMIC)");
WRITE(" ");WRITE(" ");WRITE(" ");
INTFIELDSIZE:=1;
WRITE("                    EIGENFUNCTION=",Q2-1);WRITE(" ");
INTFIELDSIZE:=2;
WRITE("                    PATH=",N1,"     OMEGA=",OM);
WRITE(" ");WRITE(" ");
WRITE("                         X              Y      ",
    "      U(X,Y)");
WRITE(" ");WRITE(" ");
FOR I:=1 STEP 2 UNTIL N-1 DO
FOR J:=1 STEP 2 UNTIL M-1 DO
    BEGIN INTFIELDSIZE:=2;
    WRITE("                    ",I DIV N,".",I REM N,"      ",
    J DIV M,".",J REM M,U(I,J));
    WRITE(" ");
    END;
IF Q2=Q1+1 THEN GO TO W ELSE STARP;
END WRIT1;
PROCEDURE STARP;
BEGIN
IF T1=50 THEN
    BEGIN
    IF Q2=1 THEN
        BEGIN Q2:=2;
        FOR I:=1 UNTIL N-1 DO
        FOR J:=1 UNTIL M-1 DO
        U(I,J):=X1(I)*Y1(J)*(1.-X1(I))*(1.-Y1(J));
        GO TO Z5;
        END;
    IF Q2=2 THEN
        BEGIN
        FOR I:=1 UNTIL N-1 DO
        FOR J:=1 UNTIL M-1 DO
            BEGIN
            U2(I,J):=U(I,J);
        U5(I,J):=0.0;
        U(I,J):=X1(I)*(1.-X1(I))*(.5-X1(I))*Y1(J)*(1.-Y1(J));
            END;
        CAT1:=K;Q2:=3;N1:=1;OM2:=5000.0;
        GO TO Z5;
        END;
    IF Q2=3 THEN
        BEGIN
        FOR I:=1 UNTIL N-1 DO
        FOR J:=1 UNTIL M-1 DO
            BEGIN
            U3(I,J):=U(I,J);
        U5(I,J):=0.0;
        U(I,J):=Y1(J)*(1.-Y1(J))*(.5-Y1(J))*X1(I)*(1.-X1(I));
```

```
            END;
        CAT2:=KE;Q2:=4;OM2:=5000.0;
        GO TO Z5;
    END;END;
Z5:
IF Q2>2 THEN GO TO Z6;
COMMENT PUT FUNCTION TO SAT. HERE;
IF T1=50 THEN
    BEGIN
    FOR I:=0 UNTIL N DO
FOR J:=0 UNTIL M DO
        BEGIN
        E1(I,J):=2.*(X1(I)**2+Y1(J)**2-X1(I)-Y1(J));
        C(I,J):=2.0*H*H*E1(I,J);
    END;END;
FOR I:=1 UNTIL M-1 DO
FOR J:=1 UNTIL M-1 DO
    BEGIN
    IF I=J THEN I1(I,J):=1.0 ELSE I1(I,J):=0.0;
    Q(I,J):=0.0;
    IF I=J THEN Q(I,J):=K1*3.0;
    IF ABS(I-J)=1 THEN Q(I,J):=-K1;
    END;
FOR J:=1 UNTIL N DO
FOR I:=1 UNTIL M-1 DO
    BEGIN
    R(J,I):=0.0;
    IF I=1 THEN R(J,I):=-2.0*K1*U(J,0);
    IF I=M-1 THEN R(J,I):=-2.0*K1*U(J,M);
    END;
FOR I:=1 UNTIL M-1 DO
FOR J:=1 UNTIL M-1 DO
  A(N,I,J):=I1(I,J);
FOR I:=1 UNTIL M-1 DO
  B(N,I):=-2.0*U(N,I);
SWITCH;
Z6:IF Q2=2 THEN SWITCH ELSE OOP;
END STARP;
PROCEDURE OOP;
COMMENT NORMALIZES U(X,Y);
BEGIN
ZNORM:=0.0;
FOR I:=0 UNTIL N DO
FOR J:=0 UNTIL M DO
    BEGIN
    ZS:=ABS(U(I,J));
    IF ZS>ZNORM THEN ZNORM:=ZS;
    END;
FOR I:=0 UNTIL N DO
FOR J:=0 UNTIL M DO U(I,J):=U(I,J)/ZNORM;
OM02:=OM2;
OM2:=1.0/ZNORM;
OM:=SQRT(OM2);
IF N1>30 THEN
    BEGIN
    WRITE("TOO MANY STEPS");
    GO TO W;
    END;
ZNORM1:=0.0;
FOR I:=1 UNTIL N-1 DO
FOR J:=1 UNTIL M-1 DO
    BEGIN
    ZS:=ABS(U(I,J)-U5(I,J));
    IF ZS>ZNORM1 THEN ZNORM1:=ZS;
    END;
IF ZNORM1<ERA THEN
    BEGIN KE:=0.0;
    FOR I:=1 UNTIL N DO
    FOR J:=1 UNTIL M DO KE:=KE+(U(I,J)*H)**2;
    WRITE;
    END;
N1:=N1+1;
```

```
FOR I:=0 UNTIL N DO
FOR J:=0 UNTIL M DO
    BEGIN
    U5(I,J):=U(I,J);
    IF V1=0 THEN E1(I,J):=-U(I,J);
    IF V1=2 THEN E1(I,J):=U(I,J);
    C(I,J):=2.*H*H*E1(I,J);
    END;
FIGURE;
END DOP;
PROCEDURE FIGURE;
BEGIN
COMMENT SOLVES FOR U AND B;
IF (T1=0) OR ((T1=50) AND (V1=0)) THEN
    BEGIN
    FOR L:=N STEP -1 UNTIL 2 DO
    FOR I:=1 UNTIL M-1 DO
        BEGIN CAT(I):=0.0;
        FOR J:=1 UNTIL M-1 DO
        CAT(I):=CAT(I)+D(L,I,J)*(B(L,J)+R(L-1,J)+C(L-1,J));
        B(L-1,I):=CAT(I);
        END;
    FOR I:=1 UNTIL N-1 DO
        BEGIN
        FOR J:=1 UNTIL M-1 DO
            BEGIN CAT(J):=0.0;
            FOR L:=1 UNTIL M-1 DO
              CAT(J):=CAT(J)+D(I+1,J,L)
              *(U(I-1,L)-(B(I+1,L)+R(I,L)+C(I,L))/2.0);
            U(I,J):=CAT(J);
    END;END;END ELSE
    BEGIN
IF TOGGLE THEN
BEGIN
    U1(0,0):=-P1(0)-P2(0);
    U1(N,0):=P3(0)-P2(N);
    U1(0,M):=-P1(M)+P4(0);
    U1(N,M):=P4(N)+P3(M);
    FOR I:=1 UNTIL M-1 DO
        BEGIN
        U1(0,I):=-P1(I)+(U(0,I+1)-2.*U(0,I)+U(0,I-1))/(H*H);
        U1(N,I):=P3(I)+(U(N,I+1)-2.*U(N,I)+U(N,I-1))/(H*H);
        END;
    FOR I:=1 UNTIL N-1 DO
        BEGIN
        U1(I,0):=-P2(I)+(U(I+1,0)-2.*U(I,0)+U(I-1,0))/(H*H);
        U1(I,M):=P4(I)+(U(I+1,M)-2.*U(I,M)+U(I-1,M))/(H*H);
        END;
    FOR I:=1 UNTIL M-1 DO B1(N,I):=-2.*U1(N,I);
    FOR J:=1 UNTIL N DO
    FOR I:=1 UNTIL M-1 DO
        BEGIN R1(J,I):=0.0;
        IF I=1 THEN R1(J,I):=-2.*U1(J,0);
        IF I=M-1 THEN R1(J,I):=-2.*U1(J,M);
        END;
TOGGLE:=FALSE;
END;
    FOR L:=N STEP -1 UNTIL 2 DO
    FOR I:=1 UNTIL M-1 DO
        BEGIN CAT(I):=0.0;
        FOR J:=1 UNTIL M-1 DO
        CAT(I):=CAT(I)+D(L,I,J)*(B1(L,J)+R1(L-1,J)+C(L-1,J));
        B1(L-1,I):=CAT(I);
        END;
    FOR I:=1 UNTIL N-1 DO
        BEGIN
        FOR J:=1 UNTIL M-1 DO
            BEGIN CAT(J):=0.0;
            FOR L:=1 UNTIL M-1 DO
            CAT(J):=CAT(J)+D(I+1,J,L)
            *(U1(I-1,L)-(B1(I+1,L)+R1(I,L)+C(I,L))/2.);
            U1(I,J):=CAT(J);
```

```
        END;END;
FOR I:=0 UNTIL N DO
    FOR J:=0 UNTIL M DO CI(I,J):=2.*H*H*U1(I,J);
    FOR L:=N STEP -1 UNTIL 2 DO
    FOR I:=1 UNTIL M-1 DO
        BEGIN CAT(I):=0.0;
        FOR J:=1 UNTIL M-1 DO
CAT(I):=CAT(I)+D(L,I,J)*(B(L,J)+R(L-1,J)+CI(L-1,J));
        B(L-1,I):=CAT(I);
        END;
    FOR I:=1 UNTIL N-1 DO
        BEGIN
        FOR J:=1 UNTIL M-1 DO
            BEGIN CAT(J):=0.0;
            FOR L:=1 UNTIL M-1 DO
            CAT(J):=CAT(J)+D(I+1,J,L)
        *(U(I-1,L)-(B(I+1,L)+R(I,L)+CI(I,L))/2.);
            U(I,J):=CAT(J);
    END;END;END;
IF O2>2 THEN
    BEGIN
    C1:=DOG1:=DOG2:=0.0;
    FOR I:=1 UNTIL N-1 DO
    FOR J:=1 UNTIL M-1 DO C1:=C1+U(I,J)*H*H;
    FOR I:=1 UNTIL N-1 DO
    FOR J:=1 UNTIL M-1 DO U(I,J):=U(I,J)-C1;
    FOR I:=1 UNTIL N-1 DO
    FOR J:=1 UNTIL M-1 DO
        BEGIN
        DOG1:=DOG1+U(I,J)*U2(I,J)*H*H;
        DOG2:=DOG2+U(I,J)*U3(I,J)*H*H;
        END;
    C2:=DOG1/CAT1;C3:=DOG2/CAT2;
    FOR I:=1 UNTIL N-1 DO
    FOR J:=1 UNTIL M-1 DO
    U(I,J):=U(I,J)-C2*U2(I,J)-C3*U3(I,J);
    END;
IF T1=50 THEN DOP;
IOCONTROL(3);
WRITE(" ");WRITE(" ");WRITE(" ");
WRITE("                              COMPUTER OUTPUT 6.(DYNAMIC)");
WRITE(" ");WRITE(" ");WRITE(" ");
WRITE("                                X                Y      ",
    "     U(X,Y)");
WRITE(" ");WRITE(" ");
FOR I:=1 STEP 2 UNTIL N-1 DO
FOR J:=1 STEP 2 UNTIL M-1 DO
    BEGIN INTFIELDSIZE:=2;
    WRITE("                              ",I DIV N,".",I REM N,"       ",
      J DIV M,".",J REM M,U(I,J));
    WRITE(" ");
    END;
GO TO W;
END FIGURE;
PROCEDURE SWITCH;
BEGIN
INTEGER L,K;
REAL DOM,AMUL,T;
FOR L:=N STEP -1 UNTIL 2 DO
    BEGIN
    S:=S-1;
    FOR I:=1 UNTIL M-1 DO
    FOR J:=1 UNTIL M-1 DO
      D(L,I,J):=Q(I,J)+A(L,I,J);
    FOR I:=1 UNTIL M-1 DO
    FOR J:=1 UNTIL M-1 DO
      D(L,I,J+M-1):=I1(I,J);
    FOR K:=1 UNTIL M-1 DO
        BEGIN
        IF K=M-1 THEN GO TO P;
        FOR I:=K+1 UNTIL M-1 DO
            BEGIN
```

```
            IF ABS(D(L,K,K))<ABS(D(L,I,K)) THEN
                BEGIN
                FOR J:=1 UNTIL 2*M-2 DO
                    BEGIN
                    T:=D(L,I,J);D(L,I,J):=D(L,K,J);
                    D(L,K,J):=T;
            END;END;END;
        P:IF ABS(D(L,K,K))<= THEN WRITER ELSE
            BEGIN
            DOM:=D(L,K,K);
            FOR J:=1 UNTIL 2*M-2 DO D(L,K,J):=D(L,K,J)/DOM;
            FOR I:=1 UNTIL M-1 DO
                BEGIN
                AMUL:=D(L,I,K);
                IF I=K THEN  ELSE
                    BEGIN
                    FOR J:=1 UNTIL 2*M-2 DO
                        BEGIN
                        D(L,I,J):=D(L,I,J)-AMUL*D(L,K,J);
        END;END;END;END;END;
    FOR I:=1 UNTIL M-1 DO
    FOR J:=1 UNTIL M-1 DO
      D(L,I,J):=D(L,I,J+M-1);
    FOR I:=1 UNTIL M-1 DO
    FOR J:=1 UNTIL M-1 DO
      A(L+1,I,J):=I1(I,J)-D(L,I,J);
    END;
IF T1=50 THEN DOP ELSE FIGURE;
END SWITCH;
STARP;
END;
END;
W:GO TO X;
Z:END.
```

COMPUTER PROGRAM 7.

DYNAMIC PROGRAMMING APPLIED TO

(1) $\nabla^2 U = -\lambda^2 U$ EQ. (1.1)

(2) $\nabla^4 U = 8.0$ EQ. (1.7)

```
BEGIN
COMMENT SOLVES PARTIAL DIFFERENTIAL EQUATIONS AND
 EIGENVALUE PROBLEMS ON A RECTANGLE BY DYNAMIC
 PROGRAMMING AND STODOLA'S METHOD.
   FORMAL  PARAMETERS:
   CM    OMEGA
   CM2   OMEGA SQUARED
   N     NUMBER OF X POSITIONS
   M     NUMBER OF Y POSITIONS
   K1    COEFF IF NEEDED
   V1    ORDER OF THE OPERATOR
          V1=0   FIRST ORDER EIGENVALUE PRB
          V1=2   SECOND ORDER EIGENVALUE PRB
   T1    TYPE OF PROGRAM RUN
             T1=0   FIRST ORDER
             T1=2   SECOND ORDER
             T1=50 EIGENVALUE
   P(I) SECOND NORMAL DERIVATIVE ON BDRY
   E1    FUNCTION TO SATISFY
   U     UNKNOWN FUNCTION
   Q1    NUMBER OF EIGENVALUES;
REAL KE,PC;
REAL E,ERA,OM,OM2,OMO2,H,K1;
REAL CAT1,CAT2,C1,C2,C3,DOG1,DOG2;
REAL ZNORM,ZS;
INTEGER N,M,N1,S,T1,V1,M1;
INTEGER Q1,Q2;
LOGICAL TOGGLE;
REAL ZNORM1;
X:READ(N,M,H,K1,T1,Q1,V1);
TOGGLE:=TRUE;
Q2:=1;
CAT1:=CAT2:=5000.0;
M1:=2*M-2;
OM2:=1000.0;
N1:=0;
ERA:=0.03;
EPA:=0.002;
E:=0.000001;
BEGIN
REAL ARRAY X1(0::N);
REAL ARRAY Y1(0::M);
REAL ARRAY U5(0::N,0::M);
REAL ARRAY U1,U2(0::N,0::M);
REAL ARRAY U3(0::N,0::M);
REAL ARRAY C1,B1,R1(0::N,0::M);
REAL ARRAY Z(0::N+4,0::M+4);
REAL ARRAY E1(0::N,0::M);
REAL ARRAY I1,Q(1::M-1,1::M-1);
REAL ARRAY U,B,R(0::N,0::M);
REAL ARRAY A(1::N,1::M,1::M);
REAL ARRAY D(1::N,0::M,0::M1);
REAL ARRAY CAT(0::M);
REAL ARRAY C(0::N,0::M);
REAL ARRAY P1,P3(0::M);
REAL ARRAY P2,P4(0::N);
FOR I:=0 UNTIL M DO READON(U(0,I));
FOR I:=0 UNTIL N DO READON(U(I,0));
FOR I:=0 UNTIL M DO READON(U(N,I));
FOR I:=0 UNTIL N DO READON(U(I,M));
FOR I:=0 UNTIL N DO X1(I):=I*H;
FOR J:=0 UNTIL M DO Y1(J):=J*H;
IF (T1>0) AND (V1=2) THEN
```

```
    BEGIN
    FOR I:=0 UNTIL M DO
    P1(I):=2.*(Y1(I)-Y1(I)**2);
    FOR I:=0 UNTIL M DO
    P3(I):=-P1(I);
    FOR I:=0 UNTIL N DO
    P2(I):=2.*(X1(I)-X1(I)**2);
    FOR I:=0 UNTIL N DO
    P4(I):=-P2(I);
    END;
FOR I:=0 UNTIL N DO
FOR J:=0 UNTIL M DO U1(I,J):=U2(I,J):=U3(I,J):=U5(I,J):=0.0;
S:=N;
BEGIN
PROCEDURE WRITER;
BEGIN WRITE("SINGULAR");
GO TO W;
END WRITER;
PROCEDURE WRIT1;
BEGIN
IOCONTROL(3);
WRITE(" ");WRITE(" ");WRITE(" ");
WRITE("                          COMPUTER OUTPUT 7.(DYNAMIC)");
WRITE(" ");WRITE(" ");WRITE(" ");
INTFIELDSIZE:=1;
WRITE("                EIGENFUNCTION=",Q2-1);WRITE(" ");
INTFIELDSIZE:=2;
WRITE("                PATH=",N1,"     OMEGA=",OM);
WRITE(" ");WRITE(" ");
WRITE("                      X                Y      ",
    "      U(X,Y)");
WRITE(" ");WRITE(" ");
FOR I:=1 STEP 2 UNTIL N-1 DO
FOR J:=1 STEP 2 UNTIL M-1 DO
    BEGIN INTFIELDSIZE:=2;
    WRITE("                ",I DIV N,".",I REM N,"       ",
      J DIV M,".",J REM M,U(I,J));
    WRITE(" ");
    END;
IF Q2=Q1+1 THEN GO TO W ELSE STARP;
END WRIT1;
PROCEDURE STARP;
BEGIN
IF T1=50 THEN
    BEGIN
    IF Q2=1 THEN
        BEGIN Q2:=2;
        FOR I:=1 UNTIL N-1 DO
        FOR J:=1 UNTIL M-1 DO
        U(I,J):=X1(I)*Y1(J)*(1.-X1(I))*(1.-Y1(J));
        GO TO Z5;
        END;
    IF Q2=2 THEN
        BEGIN
        FOR I:=1 UNTIL N-1 DO
        FOR J:=1 UNTIL M-1 DO
            BEGIN
            U2(I,J):=U(I,J);
        U5(I,J):=0.0;
        U(I,J):=X1(I)*(1.-X1(I))*(.5-X1(I))*Y1(J)*(1.-Y1(J));
            END;
        CAT1:=K5;Q2:=3;N1:=1;OM2:=5000.0;
        GO TO Z5;
        END;
    IF Q2=3 THEN
        BEGIN
        FOR I:=1 UNTIL N-1 DO
        FOR J:=1 UNTIL M-1 DO
            BEGIN
            U3(I,J):=U(I,J);
        U5(I,J):=0.0;
        U(I,J):=Y1(J)*(1.-Y1(J))*(.5-Y1(J))*X1(I)*(1.-X1(I));
```

```
            END;
        CAT2:=KE;Q2:=4;OM2:=5000.0;
        GO TO Z5;
    END;END;
Z5:
IF Q2>2 THEN GO TO Z6;
COMMENT PUT FUNCTION TO SAT. HERE;
IF T1¬=50 THEN
    BEGIN
    FOR I:=0 UNTIL N DO
FOR J:=0 UNTIL M DO
        BEGIN
        E1(I,J):=8.0;
        C(I,J):=2.0*H*H*E1(I,J);
    END;END;
FOR I:=1 UNTIL M-1 DO
FOR J:=1 UNTIL M-1 DO
    BEGIN
    IF I=J THEN I1(I,J):=1.0 ELSE I1(I,J):=0.0;
    Q(I,J):=0.0;
    IF I=J THEN Q(I,J):=K1*3.0;
    IF ABS(I-J)=1 THEN Q(I,J):=-K1;
    END;
FOR J:=1 UNTIL N DO
FOR I:=1 UNTIL M-1 DO
    BEGIN
    R(J,I):=0.0;
    IF I=1 THEN R(J,I):=-2.0*K1*U(J,0);
    IF I=M-1 THEN P(J,I):=-2.0*K1*U(J,M);
    END;
FOR I:=1 UNTIL M-1 DO
FOR J:=1 UNTIL M-1 DO
  A(N,I,J):=I1(I,J);
FOR I:=1 UNTIL M-1 DO
  B(N,I):=-2.0*U(N,I);
SWITCH;
Z6:IF Q2=2 THEN SWITCH ELSE DOP;
END STARP;
PROCEDURE DOP;
COMMENT NORMALIZES U(X,Y);
BEGIN
ZNORM:=0.0;
FOR I:=0 UNTIL N DO
FOR J:=0 UNTIL M DO
    BEGIN
    ZS:=ABS(U(I,J));
    IF ZS>ZNORM THEN ZNORM:=ZS;
    END;
FOR I:=0 UNTIL N DO
FOR J:=0 UNTIL M DO U(I,J):=U(I,J)/ZNORM;
OMO2:=OM2;
OM2:=1.0/ZNORM;
OM:=SQRT(OM2);
IF N1>30 THEN
    BEGIN
    WRITE("TOO MANY STEPS");
    GO TO W;
    END;
ZNORM1:=0.0;
FOR I:=1 UNTIL N-1 DO
FOR J:=1 UNTIL M-1 DO
    BEGIN
    ZS:=ABS(U(I,J)-U5(I,J));
    IF ZS>ZNORM1 THEN ZNORM1:=ZS;
    END;
IF ZNORM1<ERA THEN
    BEGIN KE:=0.0;
    FOR I:=1 UNTIL N DO
    FOR J:=1 UNTIL M DO KE:=KE+(U(I,J)*H)**2;
    WRITI;
    END;
N1:=N1+1;
```

```
FOR I:=0 UNTIL N DO
FOR J:=0 UNTIL M DO
    BEGIN
    U5(I,J):=U(I,J);
    IF V1=0 THEN E1(I,J):=-U(I,J);
    IF V1=2 THEN E1(I,J):=U(I,J);
    C(I,J):=2.*H*H*E1(I,J);
    END;
FIGURE;
END DOP;
PROCEDURE FIGURE;
BEGIN
COMMENT SOLVES FOR U AND B;
IF (T1=0) OR ((T1=50) AND (V1=0)) THEN
    BEGIN
    FOR L:=N STEP -1 UNTIL 2 DO
    FOR I:=1 UNTIL M-1 DO
        BEGIN CAT(I):=0.0;
        FOR J:=1 UNTIL M-1 DO
        CAT(I):=CAT(I)+D(L,I,J)*(B(L,J)+R(L-1,J)+C(L-1,J));
        B(L-1,I):=CAT(I);
        END;
    FOR I:=1 UNTIL N-1 DO
        BEGIN
        FOR J:=1 UNTIL M-1 DO
            BEGIN CAT(J):=0.0;
            FOR L:=1 UNTIL M-1 DO
              CAT(J):=CAT(J)+D(I+1,J,L)
              *(U(I-1,L)-(B(I+1,L)+R(I,L)+C(I,L))/2.0);
            U(I,J):=CAT(J);
    END;END;END ELSE
    BEGIN
IF TOGGLE THEN
BEGIN
    U1(0,0):=-P1(0)-P2(0);
    U1(N,0):=P3(0)-P2(M);
    U1(0,M):=-P1(M)+P4(0);
    U1(N,M):=P4(N)+P3(M);
    FOR I:=1 UNTIL M-1 DO
        BEGIN
        U1(0,I):=-P1(I)+(U(0,I+1)-2.*U(0,I)+U(0,I-1))/(H*H);
        U1(N,I):=P3(I)+(U(N,I+1)-2.*U(N,I)+U(N,I-1))/(H*H);
        END;
    FOR I:=1 UNTIL N-1 DO
        BEGIN
        U1(I,0):=-P2(I)+(U(I+1,0)-2.*U(I,0)+U(I-1,0))/(H*H);
        U1(I,M):=P4(I)+(U(I+1,M)-2.*U(I,M)+U(I-1,M))/(H*H);
        END;
    FOR I:=1 UNTIL M-1 DO B1(N,I):=-2.*U1(N,I);
    FOR J:=1 UNTIL N DO
    FOR I:=1 UNTIL M-1 DO
        BEGIN P1(J,I):=0.0;
        IF I=1 THEN B1(J,I):=-2.*U1(J,0);
        IF I=M-1 THEN B1(J,I):=-2.*U1(J,M);
        END;
TOGGLE:=FALSE;
END;
    FOR L:=N STEP -1 UNTIL 2 DO
    FOR I:=1 UNTIL M-1 DO
        BEGIN CAT(I):=0.0;
        FOR J:=1 UNTIL M-1 DO
        CAT(I):=CAT(I)+D(L,I,J)*(B1(L,J)+R1(L-1,J)+C(L-1,J));
        B1(L-1,I):=CAT(I);
        END;
    FOR I:=1 UNTIL N-1 DO
        BEGIN
        FOR J:=1 UNTIL M-1 DO
            BEGIN CAT(J):=0.0;
            FOR L:=1 UNTIL M-1 DO
            CAT(J):=CAT(J)+D(I+1,J,L)
            *(U1(I-1,L)-(B1(I+1,L)+R1(I,L)+C(I,L))/2.);
            U1(I,J):=CAT(J);
```

```
        END;END;
FOR I:=0 UNTIL N DO
    FOR J:=0 UNTIL M DO CI(I,J):=2.*H*H*U1(I,J);
    FOR L:=N STEP -1 UNTIL 2 DO
    FOR I:=1 UNTIL M-1 DO
        BEGIN CAT(I):=0.0;
        FOR J:=1 UNTIL M-1 DO
CAT(I):=CAT(I)+D(L,I,J)*(B(L,J)+R(L-1,J)+CI(L-1,J));
        B(L-1,I):=CAT(I);
        END;
    FOR I:=1 UNTIL N-1 DO
        BEGIN
        FOR J:=1 UNTIL M-1 DO
            BEGIN CAT(J):=0.0;
            FOR L:=1 UNTIL M-1 DO
            CAT(J):=CAT(J)+D(I+1,J,L)
        *(U(I-1,L)-(B(I+1,L)+R(I,L)+CI(I,L))/2.);
            U(I,J):=CAT(J);
    END;END;END;
IF C2>2 THEN
    BEGIN
    C1:=DOG1:=DOG2:=0.0;
    FOR I:=1 UNTIL N-1 DO
    FOR J:=1 UNTIL M-1 DO C1:=C1+U(I,J)*H*H;
    FOR I:=1 UNTIL N-1 DO
    FOR J:=1 UNTIL M-1 DO U(I,J):=U(I,J)-C1;
    FOR I:=1 UNTIL N-1 DO
    FOR J:=1 UNTIL M-1 DO
        BEGIN
        DOG1:=DOG1+U(I,J)*U2(I,J)*H*H;
        DOG2:=DOG2+U(I,J)*U3(I,J)*H*H;
        END;
    C2:=DOG1/CAT1;C3:=DOG2/CAT2;
    FOR I:=1 UNTIL N-1 DO
    FOR J:=1 UNTIL M-1 DO
    U(I,J):=U(I,J)-C2*U2(I,J)-C3*U3(I,J);
    END;
IF T1=50 THEN DOP;
IOCONTROL(3);
WRITE(" ");WRITE(" ");WRITE(" ");
WRITE("                    COMPUTER OUTPUT 8.(DYNAMIC)");
WRITE(" ");WRITE(" ");WRITE(" ");
WRITE("                              X              Y      ",
    "      U(X,Y)");
WRITE(" ");WRITE(" ");
FOR I:=1 STEP 2 UNTIL N-1 DO
FOR J:=1 STEP 2 UNTIL M-1 DO
    BEGIN INTFIELDSIZE:=2;
    WRITE("                              ",I DIV N,".",I REM N,"       ",
      J DIV M,".",J REM M,U(I,J));
    WRITE(" ");
    END;
GO TO W;
END FIGURE;
PROCEDURE SWITCH;
BEGIN
INTEGER L,K;
REAL DOM,AMUL,T;
FOR L:=N STEP -1 UNTIL 2 DO
    BEGIN
    S:=S-1;
    FOR I:=1 UNTIL M-1 DO
    FOR J:=1 UNTIL M-1 DO
      D(L,I,J):=Q(I,J)+A(L,I,J);
    FOR I:=1 UNTIL M-1 DO
    FOR J:=1 UNTIL M-1 DO
      D(L,I,J+M-1):=I1(I,J);
    FOR K:=1 UNTIL M-1 DO
        BEGIN
        IF K=M-1 THEN GO TO P;
        FOR I:=K+1 UNTIL M-1 DO
            BEGIN
```

```
                IF ABS(D(L,K,K))<ABS(D(L,I,K)) THEN
                    BEGIN
                    FOR J:=1 UNTIL 2*M-2 DO
                        BEGIN
                        T:=D(L,I,J);D(L,I,J):=D(L,K,J);
                        D(L,K,J):=T;
                ENC;END;END;
            P:IF ABS(D(L,K,K))<E THEN WRITER ELSE
                BEGIN
                DOM:=D(L,K,K);
                FOR J:=1 UNTIL 2*M-2 DO D(L,K,J):=D(L,K,J)/DOM;
                FOR I:=1 UNTIL M-1 DO
                    BEGIN
                    AMUL:=D(L,I,K);
                    IF I=K THEN ELSE
                        BEGIN
                        FOR J:=1 UNTIL 2*M-2 DO
                            BEGIN
                            D(L,I,J):=D(L,I,J)-AMUL*D(L,K,J);
        END;END;END;END;END;
    FOR I:=1 UNTIL M-1 DO
    FOR J:=1 UNTIL M-1 DO
      D(L,I,J):=D(L,I,J+M-1);
    FOR I:=1 UNTIL M-1 DO
    FOR J:=1 UNTIL M-1 DO
      A(L-1,I,J):=I1(I,J)-D(L,I,J);
    END;
IF T1=50 THEN DOP ELSE FIGURE;
END SWITCH;
STARP;
END;
END;
W:GO TO X;
Z:END.
```

COMPUTER PROGRAM 8.

GALERKIN'S METHOD APPLIED TO

$\nabla^2 U = 2(x^2+y^2-x-y)$ EQ. (1.5)

```
BEGIN
COMMENT SOLVES PARTIAL DIFFERENTIAL EQUATIONS OF
 THE FORM L(U)=F BY GALERKIN'S METHOD ON ANY DOMAIN.
   FORMAL PARAMETERS:
    N       NUMBER OF EQUATIONS
    M2      MAX. NUMBER OF X POSITIONS
    M3      MAX. NUMBER OF Y POSITIONS
    N2      ARRAY OF LOWER Y POSITIONS
    N3      ARRAY OF UPPER Y POSITIONS
    F       KNOWN FUNCTION
    U       UNKNOWN FUNCTION
    D(I)    ESTIMATING FUNCTIONS
    D1(I)   L(ESTIMATING FUNCTIONS)
    A(I)    COEF. OF ESTIMATING FUNCTIONS;
REAL H,E,DOG,CAT,DEV,AMUL,T,DOM;
REAL H1;
REAL H2,H3;
INTEGER M2,M3;
INTEGER N,M,R;
E:=0.000001;P:=0;DEV:=1.0;
READ(N,M2,M3,H2,H3);
H1:=H2*H3;
BEGIN
REAL ARRAY X(0::M2);
REAL ARRAY Y(0::M3);
REAL ARRAY B(1::N,1::N+1);
REAL ARRAY A,C(1::N);
REAL ARRAY F,U(0::M2,0::M3);
REAL ARRAY D,D1(1::N,0::M2,0::M3);
INTEGER ARRAY N2,N3(0::M2);
FOR I:=0 UNTIL M2 DO X(I):=I*H2;
FOR I:=0 UNTIL M3 DO Y(I):=I*H3;
FOR I:=0 UNTIL M2 DO READON(N2(I));
FOR I:=0 UNTIL M2 DO READON(N3(I));
FOR I:=0 UNTIL M2 DO
FOR J:=0 UNTIL M3 DO
    BEGIN
    F(I,J):=0.0;
    U(I,J):=0.0;
    FOR L:=1 UNTIL N DO
      D(L,I,J):=D1(L,I,J):=0.0;
    END;
 BEGIN COMMENT SOLVE HERE;
 PROCEDURE WRITER;
 BEGIN WRITE("SINGULAR");GO TO 0;
 END WRITER;
 PROCEDURE PERFORM;
 BEGIN COMMENT GET U(I,J) AND A(I);
 WRITE(" ");WRITE(" ");WRITE("                    COMPUTER OUTPUT 10.(GALERKIN)")
 WRITE("
 WRITE(" ");WRITE("                VALUES OF C(I) ARE");WRITE(" ");
 WRITE("
 WRITE(" ");
 FOR I:=1 UNTIL N DO
    BEGIN
    A(I):=B(I,N+1);
    INTFIELDSIZE:=1;                    C",I,"=",A(I));WRITE(" ");
    WRITE("
    END;
 FOR J:=0 UNTIL M3-1 DO
 BEGIN
 FOR I:=0 UNTIL M2-1 DO
```

```
    BEGIN DOG:=0.0;
    FOR L:=1 UNTIL N DO DOG:=DOG+A(L)*D(L,I,J);
    U(I,J):=DOG;
END;END;
WRITE(" ");WRITE(" ");WRITE(" ");
WRITE("                    X                Y          ",
    " U(X,Y)");
WRITE(" ");WRITE(" ");
FOR I:=1 STEP 2 UNTIL M2-1 DO
FOR J:=1 STEP 2 UNTIL M3-1 DO
    BEGIN INTFIELDSIZE:=2;
    WRITE("                    ",I DIV M2,".",I REM M2,
        "          ",J DIV M3,".",J REM M3,U(I,J));
    WRITE(" ");
    END;
GO TO Q;
END PERFORM;
PROCEDURE SWITCH;
BEGIN COMMENT SOLVES FOR A(I) COEFF;
INTEGER N1,M1;
M1:=N;N1:=N+1;
FOR I:=1 UNTIL M1 DO B(I,M1+1):=C(I);
FOR K:=1 UNTIL M1 DO
    BEGIN IF K=M1 THEN GO TO P;
    FOR I:=K+1 UNTIL M1 DO
        BEGIN IF ABS(B(K,K))<ABS(B(I,K)) THEN
            BEGIN R:=R+1;
            FOR J:=1 UNTIL N1 DO
                BEGIN T:=B(I,J);B(I,J):=B(K,J);B(K,J):=T;END;
        END;END;
    P:IF ABS(B(K,K))<E THEN WRITER ELSE
        BEGIN DEV:=DEV*B(K,K);DOM:=B(K,K);
        FOR J:=1 UNTIL N1 DO B(K,J):=B(K,J)/DOM;
        FOR I:=1 UNTIL M1 DO
            BEGIN AMUL:=B(I,K);
            IF I=K THEN  ELSE
                BEGIN
                FOR J:=1 UNTIL N1 DO
                BEGIN B(I,J):=B(I,J)-AMUL*B(K,J);
    END;END;END;END;END;
PERFORM;
END SWITCH;
COMMENT GUESSED FUNCTIONS D(I) HERE;
FOR J:=0 UNTIL M2 DO
FOR K:=N2(J) UNTIL N3(J) DO
    BEGIN
    F(J,K):=2.*(X(J)**2+Y(K)**2-X(J)-Y(K));
    D(1,J,K):=(X(J)**2-X(J)**3)*(Y(K)-Y(K)**2);
    D(2,J,K):=(X(J)-X(J)**2)*(Y(K)**2-Y(K)**3);
    D(3,J,K):=(X(J)**2-X(J)**3)*(Y(K)**2-Y(K)**3);
    D1(1,J,K):=(2.-6.*X(J))*(Y(K)-Y(K)**2)+
        2.*(X(J)**3-X(J)**2);
    D1(2,J,K):=(2.-6.*Y(K))*(X(J)-X(J)**2)+
        2.*(Y(K)**3-Y(K)**2);
    D1(3,J,K):=(2.-6.*X(J))*(Y(K)**2-Y(K)**3)+
        (2.-6.*Y(K))*(X(J)**2-X(J)**3);
    END;
COMMENT SOLVE FOR INTEGRALS F*D AND LD*D;
FOR I:=1 UNTIL N DO
    BEGIN DOG:=0.0;
    FOR J:=1 UNTIL M2 DO
    FOR K:=N2(J)+1 UNTIL N3(J) DO
    DOG:=DOG+(F(J,K)*D(I,J,K)*H1);
    C(I):=DOG;
    FOR J:=1 UNTIL N DO
        BEGIN CAT:=0.0;
        FOR K:=1 UNTIL M2 DO
        FOR L:=N2(K)+1 UNTIL N3(K) DO
        CAT:=CAT+(D1(J,K,L)*D(I,K,L)*H1);
        B(I,J):=CAT;
        END;
    IF I=N THEN SWITCH;
```

```
        END;
    END;END;
0:  END.
```

BIBLIOGRAPHY

1. Angle, E. S., "discrete Invariant Imbedding and Elliptic Boundary-Value Problems over Irregular Regions," Journal of Mathematical Analysis and Applications, v. 23, p. 471-481, 1968.

2. Angle, E. S., "A Building Block Technique for Elliptic Boundary-Value Problems over Irregular Regions," Journal of Mathematical Analysis and Applications, v. 26, p. 75-81, 1969.

3. Bowdler, Hilary, Martin, R. S., Reinsch, C., Wilkinson, J. H., "The QR and QL Algorithms for Symmetric Matricies," Numerische Mathematik, v. 11, p. 293-306, 1968.

4. Dettman, J. W., Mathematical Methods in Physics and Engineering, 2d. ed., p. 162-168, McGraw-Hill, 1969.

5. Forray, M. J., Variational Calculus in Science and Engineering, p. 157-204, McGraw-Hill, 1968.

6. Kantorovich, L. V., Krylov, v. I., Approximate Methods of Higher Analysis, p. 258-303, Interscience Publishers, 1958.

7. Martin, R. S., Reinsch, C., Wilkinson, J. H., "Householder's Tridiagonalization of a Symmetric Matrix," Numerische Mathematik, v. 11, p. 181-195, 1968.

8. Nemhauser, G. L., Introduction to Dynamic Programming, p. 227-242, Wiley, 1966.

9. Stakgold, I., Boundary Value Problems of Mathematical Physics, v. 1, p. 228-245, MacMillian, 1969.

10. University of Southern California USCEE-237, Dynamic Programming and Linear Partial Differential Equations, by E. S. Angel, January 1968.